EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

Data Management Division

DFS Software

NG/AMS

Next Generation Archive Management System

User Manual

Doc. No.: VLT-MAN-ESO-19400-2739

Issue: 1

Date: 05/03/2002

DRAFT - FOR INTERNAL USE ONLY

	Name	Date	Signature
Prepared:	J.Knudstrup	05/03/2002	•
Approved:	A.Wicenec	/ /2002	•
Released:	M.Peron/P.Quinn	/ /2002	•

CHANGE RECORD

ISSUE	DATE	SECTION/PAGE AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
1/Preperation 1	29.01.2002	All	First issue.
1/Preparation 2	05.03.2002	All	Added comment after internal review.

TABLE OF CONTENTS

1 About this Guide

1.1 Purpose & Scope

This document is the user's manual for the Next Generation Archive Management System (NG/AMS). NG/AMS is the SW for the Next Generation Archive System **[2]**. It is in charge of handling the registration of storage disks and to follow their movements. In addition it handles the archiving and registering of data files. The system also provides features for retrieving and processing data. Other services are provided as well.

This manual contains all the information needed for installing, configuring and operating NG/AMS. It is also described how to enhance the system with new features by adding various types of plug-ins. These plug-ins are small Python functions with a specific interface and a specific set of tasks.

The audience of this document is high-level users who wish to perform archiving and retrieval of data files into/from the NGAS system. However, also more advanced users who needs to tune and adapt the system by changing the configuration. Finally support for the very advanced user is provided. The latter type of user is the user who adds or changes functionality of the system by providing new plug-ins or changing existing ones.

1.2 How to Read this Manual

The intention with this manual is not to provide a 'book' to read sequentially chapter by chapter. For the user unknown to NG/AMS it is recommended to read this chapter and chapter 2 to get an overview of the manual and of NG/AMS and its features. For more specific issues it is suggested to check the index or the table of contents and read the referenced sections in connection with these issues.

The following conventions are used in this manual:

Item	*Description*
`<text>`	Courier font for examples of source code files and configuration files.
"<name>"	Names of SW modules , classes, methods, functions, etc., are contained in quotes.
<element>[.<element>]	Reference to an XML element.
<element>:<attribute>	Used to refer to a specific attribute in an XML document, e.g.: "NgamsCfg:Ngams:CentralUnit".
CFG: <configuration component>	Reference to an element/attribute in the NG/AMS configuration.
DB: <DB column>	Refers to a DB column.

Some sections are dedicated to the more advanced users of NG/AMS. These sections are marked with "***EXPERT:***". A 'normal user' may want to skip these sections.

The last chapter (19) contains a quick reference to the commands supported by NG/AMS.

1.3 How to Get Help or Report Problems with NG/AMS or this Manual

In case problems are encountered using the NG/AMS, bug reports should be submitted by email to:

ngast@eso.org

This also goes for questions and other assistance needed in connection with the usage and enhancement of the system

1.4 Disclaimer

Although great efforts have been invested in designing robust interfaces for the NG/AMS SW e.g. when it comes to the HTTP communication procotol, various XML document formats, and the interfaces of the APIs provided, it should be noted that NG/AMS is still in an early phase, and minor changes may have to be introduced in the various interfaces. It will however be attempted to limit the amount of such changes to an absolute.

1.5 Reference Documents

The following documents contain additional information and are referenced in the text:

Reference	*Document Number*	*Issue*	*Date*	*Title*
[1]	VLT-SPE-ESO-19400-2534	1	22.06.2001	"DFS Software, Next Generation/Archive Management System", Design Description, J.Knudstrup.
[2]	http://archive.eso.org/NGAST	-	-	Next Generation Archive Systems Technologies, A.Wicenec.

1.6 Acronyms

The following abbreviations and acronyms are used in this document:

DB	Database
DHPI	Data Handling Plug-In
DPPI	Data Processing Plug-In
DTD	Document Type Definition
HDD	Hard Disk Drive
N/A	Not Applicable
NGAS	Next Generation Archive System
NGAS DB	NGAS (Data Holding) DB
NGAST	Next Generation Archive System Technologies
NG/AMS	NGAS Archive Management System
OS	Operating System
SW	Software
XML	Extensible Markup Language

1.7 Glossary

The following glossary is used in this document:

Archive Request	Request from a client of the NG/AMS Server to have a file archived.
Back-Log Buffering	Back-Log Buffering can be carried out by the NG/AMS Server if an error occurs, which makes it impossible to archive the file at that moment. The file will thus be stored temporarily in the Back-Log Buffer Area. The NG/AMS Server (Janitor Thread) will attempt at a later stage to handle the file.
Bad File	A Bad File is a file that could not be accepted for archiving by the NG/AMS system. I.e. it was rejected by the DHPI handling this file type. This could e.g. be due to a wrong expected size of a FITS file.
Bad Files Directory **Bad Files Area**	Area on the disk where files which are mal-formed are stored. There is a Global Bad File Directory on one of the system disks on each NGAS System. Apart from that, there is a Bad Files Directory on each archive disk installed.

Logical Name	A 'human' readable name which is used when referring to disks. The disks will typically be labelled with the Logical Name. Should be unique, although this may not be guaranteed as for the Disk ID.
Disk, Hard Disk Drive	In the context of NG/AMS the term disk refers to a random access storage device, which can be mounted under UNIX, and which has a file system, created on it.
Main (Data) File	The copy of the data file stored on the Main Storage Area.
Main (Storage) Area	The array of HDDs in an NGAS System which, when filled with data are send to the Central Archive Facility.
NG/AMS Server	Is the central process of the NGAS System. It receives the data file from the Data Providers, invokes the appropriate DHPI on the data, and ingests the information about the data in the NGAS DB.
Processing Area	Directory used to store temporary copies of files to be processed and other temporary files created during processing.
Replication (Data) File	The copy of a data file, which is stored in the Replication Area.
Replication (Storage) Area	The array of HDDs that contains the replicas of the data stored on the disks in the Main Storage Area.
Staging Area	A storage location (directory) used to temporarily store data files being handled. NG/AMS e.g., uses a Staging Area on each target disk, for receiving data files before moving the files to their final location.
Storage Set	A storage unit, which consists of either one or two disks on which data is archived.

2 Overview

In this chapter the basic concepts of NGAS and NG/AMS are described. An overview of the NG/AMS is given, as well as a description of the various fundamental features and services provided by NG/AMS. This chapter provides a somewhat high-level description of the most important features and services. More in-depths descriptions can be found in the subsequent chapters.

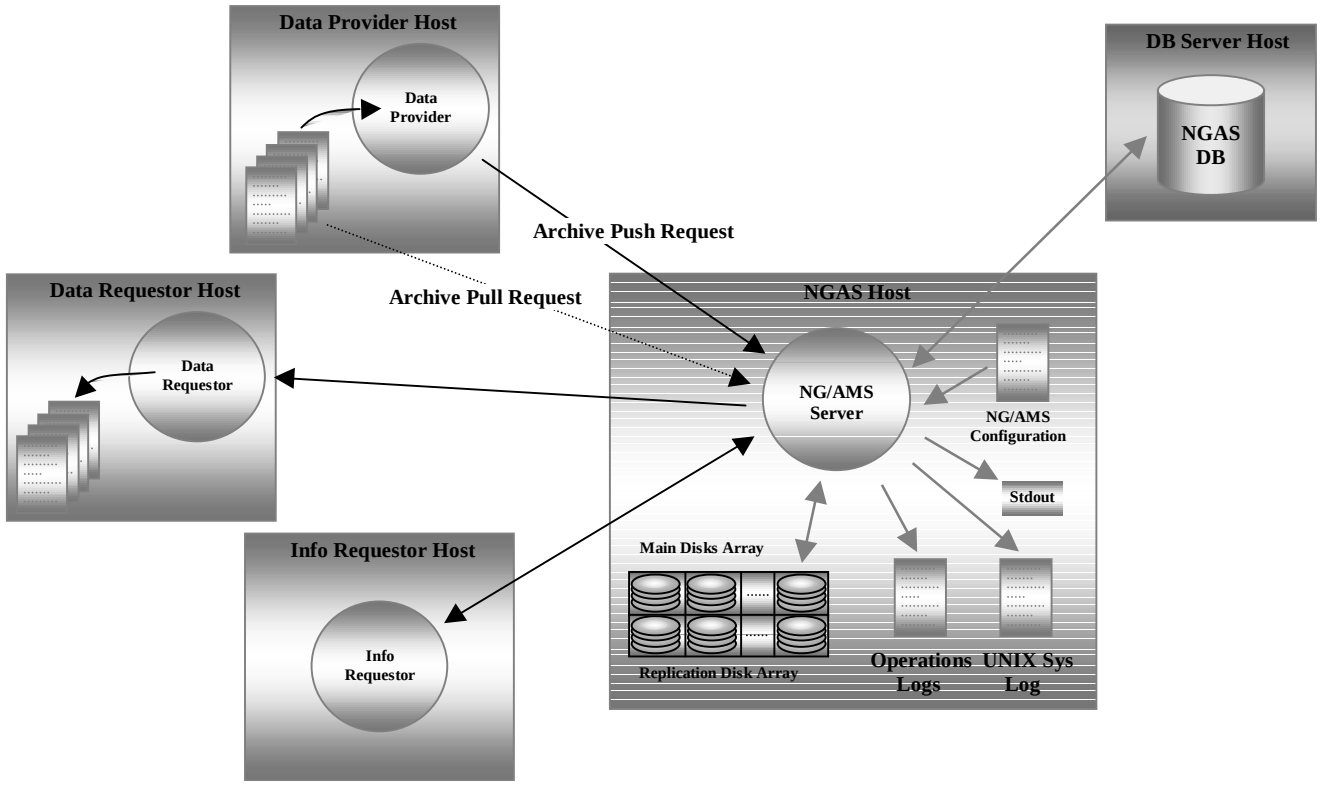


Figure 2.1: Example environment of the NG/AMS Server.

2.1 The NGAS Concept

The concept of NGAS is to use random access hard disk drives for storing data in a fast manner. In addition the price per storage unit is relatively low compared to other solutions.

Storing data on HDDs has several advantages over the present scheme whereby CD-ROM and DVD discs are used to store the data. Some advantages are:

- The archiving of data files can be carried out very fast.
- Data is online as soon as it has been archived.
- It is not necessary to store data in an intermediate location and to generate later the final media.
- The processing power of the computers hosting the disks can be used to process the data both during archiving and while retrieving data.
- In general an NGAS system requires much less manual intervention.

The NGAS is based on having standard PCs with Linux installed on them, and with a set of HDD sliders in which HDDs can be inserted and removed easily. It is foreseen to have an NGAS host at the telescope site where it will receive the data produced by the instrument(s). As soon as a disk is full, it will be send to a central archive site where it will be installed in a free slot in an NGAS host. The data is immediately online as soon as the NG/AMS has 'recognized' it. NG/AMS can produce a replication disk so that a back up of the data is available. For more information about the NGAS system check out **[2]** and the links found at this site.

2.2 Services & Features

Some of the main services and features provided by the NG/AMS SW are:

- **Multithreaded Server:** The NG/AMS Server is using threads when handling requests from clients. This means that it is capable of handling several requests simultaneously.
- **HTTP Protocol:** The communication interface of NG/AMS is based on the standard HTTP protocol. This makes it easy and possible to access the server from various clients. It is even possible to the server interaction from a WEB browser.
- **Flexible Adaptation via Configuration File:** The NG/AMS Server is configuring itself at start up, based on a large number of configuration parameters defined in the NG/AMS Configuration, which is an XML document. This makes it possible to adapt the server for specific contexts in a flexible way.
- **Adding of Specific Behavior Based on Plug-In Concept:** NG/AMS is implemented in a way so that only the kernel/general functionality is implemented (hardcoded) into the server SW. All the specific, context specific features are provided based on a plug-in scheme making it possible to adapt the server in a very flexible way. As an example, the specific handling of data during archiving, must be handled by a plug-in provided for each type of data.
- **State Management:** The NG/AMS Server maintains a State/Sub-State scheme to make it possible to restrict the services provided according to the 'condition' of the server.
- **XML Information Exchange:** All information sent back from the server (status messages) is based on XML.
- **APIs for C and Python:** APIs for communicating with the server are provided for applications written in C and Python.
- **SW Modularity/Reusage:** The NG/AMS SW is implemented as a number of classes and library functions, which can be used to build dedicated servers and other applications if needed.
- **Command Line Utilities:** Two command line utilities for communicating with the NG/AMS Server is provided. These are based on the NG/AMS C and Python APIs.
- **Data File Archiving via Push/Pull Technique:** Efficient archiving of data files is provided based on an Archive Pull Technique where NG/AMS picks up files given by a URI, and based on an Archive Push Technique, where the data provider writes the data to the server.
- **Data File Retrieval & Processing:** NG/AMS provides a scheme for transparent access to data. Based on the information in the NGAS DB, a contacted NG/AMS Server can locate the data requested by the user and provide this to the user by acting as a proxy (transparent data access). It can also send back HTTP redirection messages to indicate to the data requestor where to find the data. The C and Python APIs handle the data access completely transparent for the client.
- **Access/Service Restriction:** It is possible to configure the enabling/disabling of some basic services. These are for the moment: Handling of Archive Requests, Handling of Retrieve Requests, and Data Processing.
- **Back-Log Buffering of Data:** In case problems occur preventing NG/AMS from archiving data, NG/AMS will Back-Log Buffer data and try to handle this at a later stage.
- **Disk Registering & Supervision:** When a disk first has been registered by NG/AMS, it will follow it movements around in the system, and always keep the DB up to date to indicate the latest status of the disk.
- **File Registering:** A number of parameters are registered for the files archived in the NGAS DB, making it possible to locate, retrieve and process these files.
- **Handling Data Replication:** NG/AMS can handle replication of data files if requested. Also the information for such replicated files is updated automatically in the NGAS DB.
- **Canalizing Data Streams:** Via the configuration file it is possible to define how NG/AMS should stream data onto the various Storage Disks available in an NGAS host.
- **Generation of Checksum:** NG/AMS generates a checksum value for each file generated. This is based on a plug-in concept so that context/data specific checksum calculation can be applied.
- **Logging:** A quite substantial set of information can be logged according to different levels on stdout, in the syslog, and in a log file.
- **Data consistency Checking Feature:** If enabled, an NG/AMS Server will run a periodic data consistency check of the data stored on the disks under its control. Via a number of parameters it is possible to adjust quite accurately how much load and how long time this task should take up.
- **Email Notification Service:** A service is provided for notifying subscribers happening via email messages about a number of different events occurring. Examples of such events are errors, disk change requests and data inconsistency reports.
- **Extendable for Usage with Various DBMS':** NG/AMS is prepared for usage with various DBMS'. For now only Sybase is supported, but this is easy to expand.
- **Information Query:** A set of various types of information can be queried via the STATUS command. This information is such as the state of the system, or information about files and disks.
- **Production of Disk Labels:** NG/AMS can produce labels for the disk cases on request. The actual SW to operate the printer must be provided in the form of a plug-in.
- **Simulation Mode:** NG/AMS provides a Simulation Mode, which makes it possible to operate the system without the availability of the actual HW, like the disk controller, disks, etc. Running in Simulation Mode, a simulated NG/AMS environment is generated on a single disk. This is useful for test and development. The Simulation Mode however, could also be used to run an NG/AMS on a 'normal' workstation for archiving data in a production system.
- **Online Documentation:** Thorough and accurate documentation contained in the Python source code of NG/AMS is provided. This makes it possible to browse the documentation online e.g. using pydoc.

The services and features listed above and described shortly, are explained in more detail in this and the following chapters. Additional services are foreseen.

2.3 Architecture

The heart of the NG/AMS is the NG/AMS Server. This is a multithreaded server based on the standard HTTP protocol. It can be seen as a dedicated WEB server. Since the server is multithreaded it is possible to issue several requests simultaneously. A number of commands are provided by NG/AMS. For more detailed information about these commands consult chapter 19. For more information about the technical details of the command interface, consult chapter 5.

NG/AMS is based on random access type of storage media, which can be mounted in UNIX style and on which a file system can be created. Although the system has been developed on UNIX it may be possible to port the SW relatively easily to other platforms supporting Python like e.g. MS-Windows.

2.4 Starting & Stopping the NG/AMS Server

The NG/AMS Server can be invoked with a number of different command line parameters. These are described in section 3.1. It is mandatory to specify an NG/AMS Configuration to be used by the NG/AMS session. How to configure the NG/AMS environment is described in chapter 4. The server can be started with the "-v" option to produce output on stdout. Normally, in a production environment, it will be started as a back-ground process, which only produces log output to the UNIX syslog and/or a Local Log File.

The server can be stopped either by sending a SIGTERM signal (15) or by sending the EXIT command, which can be issued when the server is in Ofline State (see also 2.5). If the server is killed with a SIGTERM signal, it will invoke internally a signal handler that cleans up the environment and shuts it down in a proper manner whereby also the System Offline Plug-In (chapter 11) is invoked. Also when issuing an EXIT command, the server invokes the proper 'clean-up procedure'. If the server is killed by a SIGKILL (9) signal, the signal handler is not invoked, and the server leaves its environment in an 'undefined' state. This also happens if the computer on which the server is running is shut down abruptly. If this happens it will be necessary to start the server subsequently with the "-force" parameter to force it to start-up. It should be possible to clean up the system by bringing it Online/Offline in the proper manner.

2.5 The NG/AMS Server States & Sub-States

The NG/AMS Server is maintaining a scheme of a State and a Sub-State, which determines which services the server can handle at a given point in time, and which indicates the 'condition' of the server. The States and Sub-States and the corresponding conditions are as follows:

State ⇩ Sub-State ⇨	Idle	Busy
Offline	This is the condition in which the NG/AMS Server enters after starting up, and when the OFFLINE command hs been issued. In this state only the STATUS command is accepted. I.e., no Archive or Retrieval Requests are handled. The EXIT command is also accepted. Latter makes the server clean up and terminate.	In this state the server is performing the transition from Offline to Online, or is preparing to exit from execution. No commands are accepted.
Online	In this state the server is ready to handle commands like ARCHIVE and RETRIEVE. In addition the OFFLINE command is accepted.	In this state the server is busy handling one or more Archiving or Data Retrieval Requests. Also the STATUS command is accepted. An OFFLINE command will be rejected.

It is possible to query the state of the server by issuing a STATUS command without parameters. The reply to a STATUS command is an XML document with the following contents:

```
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-02-12T18:22:54.819" HostId="arcus2" Message="Successfully handled command STATUS"
          State="ONLINE" Status="SUCCESS" SubState="IDLE" Version="v1.5/2002-02-12T10:52:10"/>
</NgamsStatus>
```

2.6 The NG/AMS Storage Media Infrastructure

The disk infrastructure used by NG/AMS is depicted in figure 2.2.

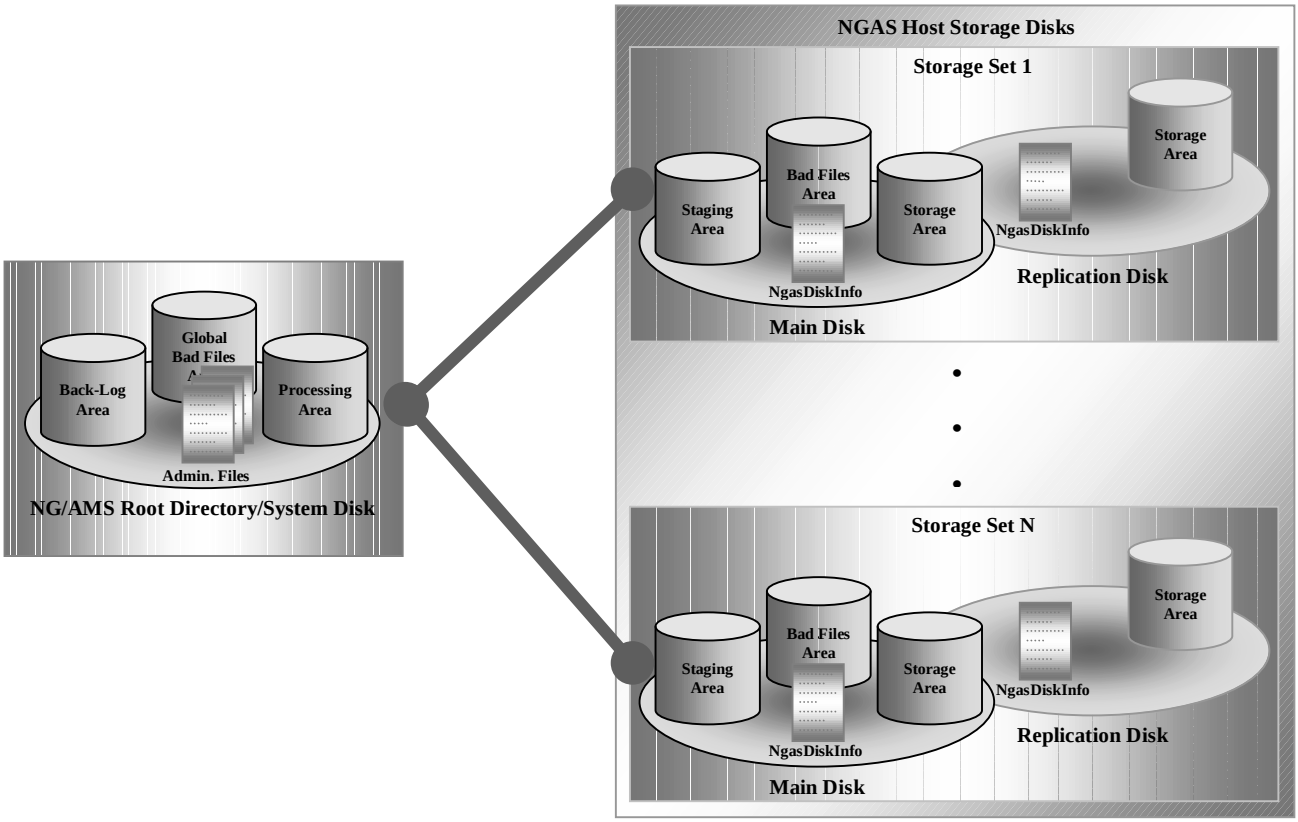


Figure 2.2: The NG/AMS 'Disk Infrastructure'.

The NG/AMS disk infrastructure, is based on a single root directory under which the Storage Disks are mounted. Under this area, NG/AMS is also storing some files for internal purposes. Among this is a file containing the PID of the NG/AMS Server process (Name: ".NGAS_<NGAS ID>".). The Back-Log and Global Bad Files Directories can be placed in a location of choice. This is done via the NG/AMS Configuration. The names of these are "<NgamsCfg.Ngams:MountRootDirectory>/bad-files" and "<NgamsCfg.Ngams:MountRootDirectory>/back-log".

The Processing Area (Directory) shown in figure 2.2, is used by NG/AMS for storing temporary files while doing file processing. The files stored in this directory will be removed by NG/AMS after the processing has finished. The name of this directory is: "<NgamsCfg.Ngams:ProcessingDirectory>/processing". Some care should be applied when allocating the location of these directories, since it may have an influence on the performance of the system. E.g., if a location for the Processing Area is chosen, which has a poor I/O performance, this may slow down the processing considerably.

The data archived on an NGAS system, is stored on Storage Sets. A Storage Set can consist of two disks, one Main Disk, and one Replication Disk. Hence it is possible to make NG/AMS carry out replication of the files being archived. This feature can also be disabled (CFG: "NgamsCfg.Ngams:Replication"). The data files archived must be stored under a single directory (referred to as Storage Area in figure 2.2) in the mount directory on the target disks. The name of this area is configurable (CFG: "NgamsCfg.FileHandling:PathPrefix"). It is up to the DHPI implementation to define the structure of the directories and files in the Storage Area. On the data disks there is also a Staging Area used by NG/AMS when receiving data files. Data is received directly onto the Main Target Disk for efficiency reasons. The name of this directory is: "<NgamsCfg.Ngams: MountRootDirectory>/<mount directory>/staging". There is only one such Staging Directory on the Main Disk. In case a file is identified as bad by the DHPI, it is stored in the (Local) Bad Files Area. The exact name of this is: "<NgamsCfg.Ngams:MountRootDirectory>/<mount directory>/bad-files". A file could be considered as bad e.g. if a checksum value for the file is found to be inconsistent. There is only a Bad Files Area on the Main Disks. Also located on the Data Disks is a file named "NgasDiskInfo". This file is an XML document that contains a resume of the information about the disk contained in the DB. An example of such a file can be found in section 16.2.

2.7 Data Classification & Handling

One of the fundamental concepts behind NG/AMS is the way data is classified and handled. This is based on the same concept as used by many WEB browsers and mail tools, namely on the mime-type of the data, which again is derived from the extension of the data files. In NG/AMS no mime-types for the data files handled are hard-coded into the SW. By means of the NG/AMS Configuration mime-types for new types of data files to be handled can be added. Note that for new types of data a corresponding DHPI must be provided. If NG/AMS encounters a data file with an unknown mime-type (not defined in the configuration) while handling an Archive Request, the request will be rejected.

It is also possible to define an arbitrary number of Data Streams, normally one per each type of data to be handled. In the data stream the following information must be defined:

Mime-Type	The mime-type indicating for which data the stream is defined.
DHPI + Parameters	The DHPI that should be used to handle the processing and archiving of the data file. In addition parameters for the DHPI can be specified in the configuration file.
Target Storage Set(s)	One or more Storage Sets, on which the data can be stored.

See chapter 4 for more information about the NG/AMS Configuration.

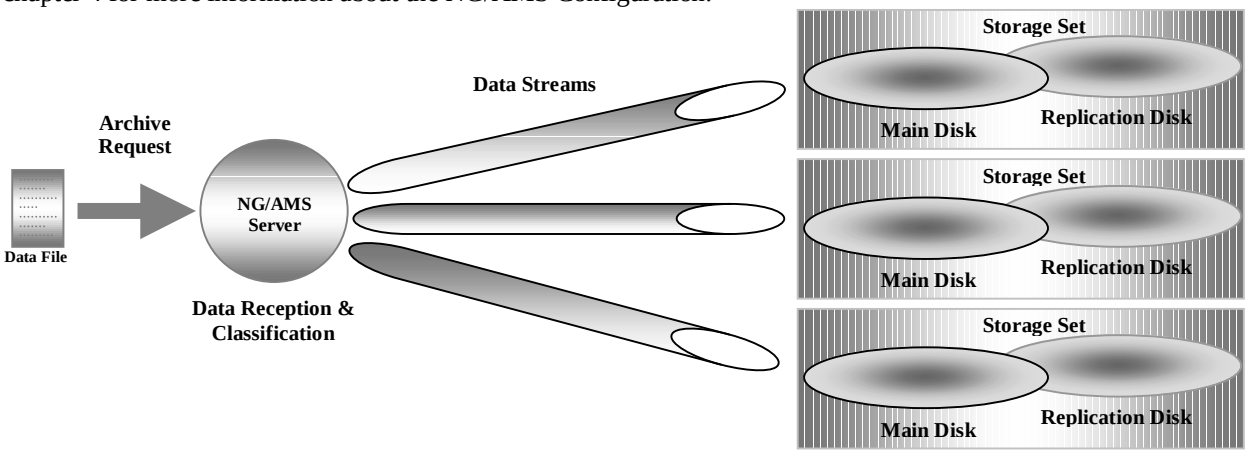


Figure 2.3: Data channeling.

A number of standard mime-type are used by NG/AMS. These are:

ngas/cmd	Used to indicate that the request to NG/AMS contains a command.
text/xml	Used by NG/AMS to indicate that a reply contains an XML document.
ngas/archive-push	Used to indicate to NG/AMS that the request is an Archive Push Command.

Using the NG/AMS APIs (see chapter 7 and 8), the user/client normally does not have to worry about this aspect.

2.8 Disk Handling/Life Cycle of a Storage Disk

In this section the various stages in the life cycle of an 'NGAS disk' are described. In the diagram in figure 2.3, a typical life cycle for an NGAS disk is shown. There might be differences for the various contexts how the actual disk handling is implemented.

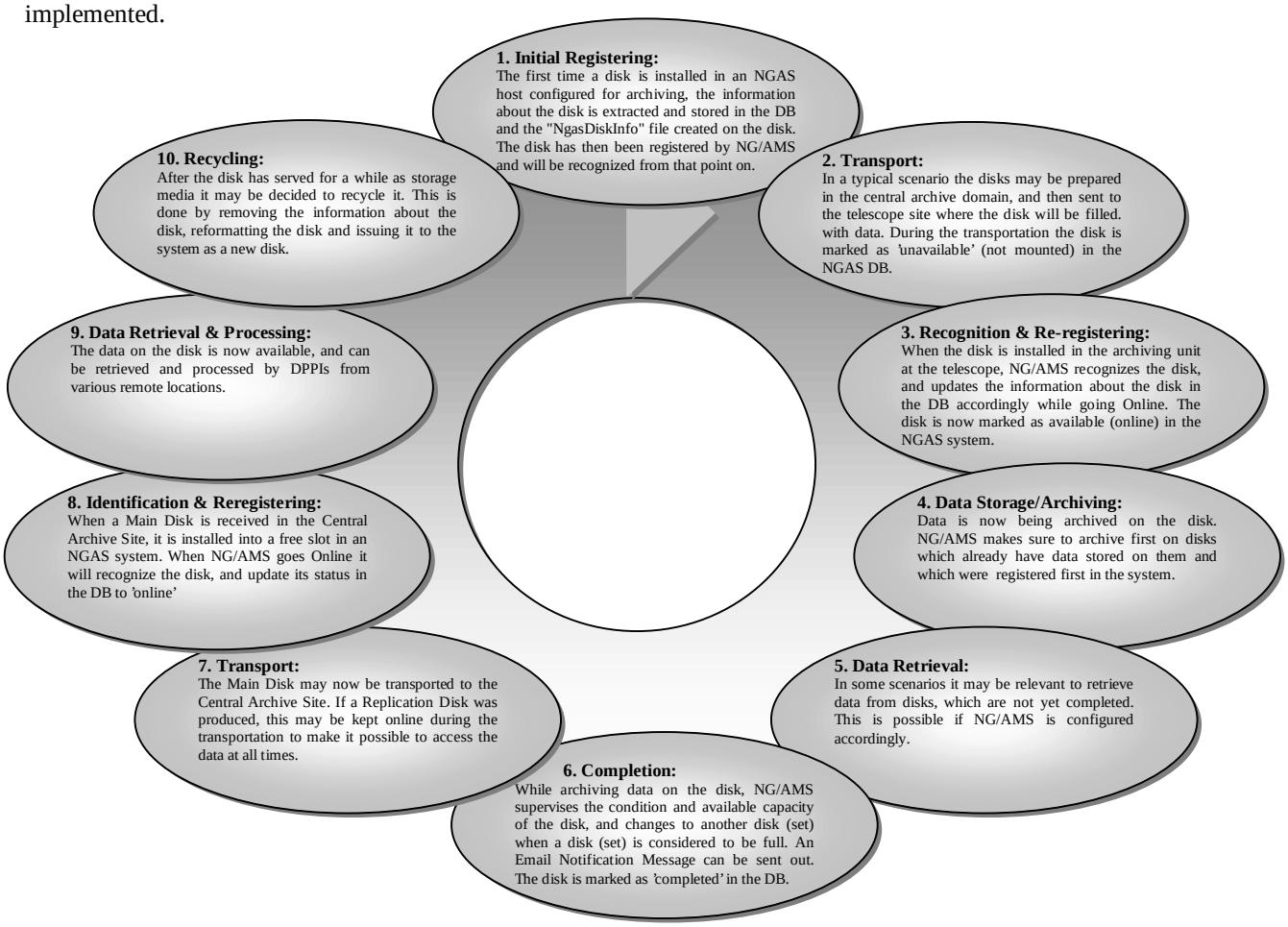


Figure 2.4: Life cycle of an NGAS disk.

A special flag in the NG/AMS Configuration, Central Unit (CFG: "NgamsCfg.Ngams:CentralUnit") is used to indicate if the NGAS host where this instance of NG/AMS is running, is part of the Central Archive. This is used to indicate that a disk is completed and 'in usage' and that the disk, the data on it and the basic, pertinent DB information should not be touched.

2.9 Data File Archiving

Archiving of files in NG/AMS is done via the ARCHIVE command. Data files can be archived either by using the Archive Push Technique or by the Archive Pull Technique. Using the former, the application archiving the file, reads in the file and sends the contents of the file in the HTTP message body. When making use of the latter technique, the client sends a URL indicating a location where the file can be picked up by NG/AMS. This URL has thus to be accessible for NG/AMS.

NG/AMS maintains a scheme of file versioning, whereby if a file with the same File ID is issued several times, the version number in the NGAS DB (DB: "ngas_files.file_version") is incremented by one. The first file with a given ID has version number 1.

In short, the handling of an Archive Request is as follows: NG/AMS receives the Archive Push or Pull request. It determines the mime-type of the data file, and from the configuration and the current disk status, which it reads from the NGAS DB, it find a suitable Target Disk for the data. The data is subsequently received into the Staging Area of the Main Target Disk. Afterwards the DHPI corresponding to the type of data in question, is invoked to do the specific handling of the data. After the DHPI has finished its processing, it returns control to NG/AMS. The DHPI has then extracted/produced

the necessary information for NG/AMS to be able to update its status in the DB and to be able to move the file to its final destination. If replication is enabled, NG/AMS also carries out this action and updates the information for the Replication File in the DB.

The ARCHIVE command is described in section 19.1. A more detailed description of the procedure executed by NG/AMS while handling an Archive Request can be found in chapter 13 (describing the DHPI).

2.10 Data File Retrieval

Archived files can be retrieved from NG/AMS using the RETRIEVE command. In the present implementation it is only possible to retrieve one file at a time. It is possible to request to have the data processed by a DPPI before NG/AMS returns it. The concept of the DPPIs is described in detail in chapter 14.

When NG/AMS receives a Retrieve Request, it checks the NGAS DB for the various instances of a data file with the given ID, which are online. Several versions may be available. The decision of which file to take is done as follows:

A list of all files with the given File ID, which are registered as being online, is retrieved from the NGAS DB. In addition, files marked to be 'ignored' are not considered.

The files are ordered according to their File Version, whereby the latest file (highest version number), gets the highest priority.

It is subsequently investigated where the instances of the given file are located. Three cases are considered: 1) Local Host - The file is stored on the NGAS host handling the request, 2) Private Network - The file is stored on an NGAS host within the private network (IP address of the format: 10.X.X.X), 3) Remote Location - The file is stored on a remote NGAS host not contained within private network. I.e., if a file with a given ID and the same version is found on several NGAS hosts, the selection criteria for which file to take is done according to the priority list described in this paragraph. I.e., if a file is found on the local host and on a remote host, the instance on the local host is taken.

When a file is found to be residing on the local host or on an NGAS host within the private network, NG/AMS always gets the file, processes it if requested and sends it back to the requestor. If the file is located on a remote host, NG/AMS will either send back an HTTP redirection response (HTTP Status Code: 303), or it can act as proxy. In the former case it is then up to the requestor to re-issue the request to the NGAS host referenced in the redirection HTTP response (see also section 5.3). In the latter case, it retrieves the file from the remote location, and subsequently sends back this file to the requestor. Whether an NG/AMS Server should act as a proxy or not, is configurable (CFG: "NgamsCfg.Ngams.ForceProxyMode", 1 = Proxy Mode).

If a file is located on a host within the private network or on a remote host, possible processing requested will be carried out on the host on which the file resides.

Methods for retrieving files in an easy manner are provided by the C and Python APIs. See the chapters 7 and 8.

2.11 Logging

As indicated in figure 2.1, a number of different types of log output can be produced by NG/AMS. These and their properties are:

Log Type	Description
Local Log File	The location of a Local Log File is defined in the NG/AMS Configuration file (CFG: "NgamsCfg.Log:LocalLogFile"). I.e., the user can decide himself where to put this file. The level (intensity) with which there is logged, can be adjusted as well (CFG: "NgamsCfg.Log:LocalLogLevel"). Note, that NG/AMS will continue to write (append logs) in the same log file. I.e., it should be considered to implement means to purge the log file[1] periodically. Examples of some entries in a Local Log File are: … `2002-02-22T14:31:43.110 [INFO] Handling HTTP request: client_address=('134.171.16.97', 32887) - method=POST - path=|ARCHIVE| - content-disposition=file_uri="SmallFile.fits"; wait="1" - content-type=ngas/archive-push - host=acngast1 - content-length=69120`

[1] A mechanism for this may be provided later within NG/AMS.

	2002-02-22T14:31:43.120 [INFO] Received command: ARCHIVE 2002-02-22T14:31:43.120 [INFO] Handling Archive Push Request ... 2002-02-22T14:31:43.120 [INFO] NGAMS_INFO_ARCHIVING_FILE:4019:INFO: Archiving file with URI: SmallFile.fits 2002-02-22T14:31:43.120 [INFO] Archiving file: SmallFile.fits with mime-type: ngas/fits ... 2002-02-22T14:31:43.210 [INFO] Plug-In handling data for file with URI: SmallFile.fits 2002-02-22T14:31:44.140 [INFO] NGAMS_INFO_FILE_ARCHIVED:4020:INFO: Successfully archived file with URI: SmallFile.fits 2002-02-22T14:31:44.140 [INFO] NGAMS_INFO_FILE_ARCHIVED:4020:INFO: Successfully archived file with URI: SmallFile.fits. Time (s): 1.01313507557 …
(UNIX) Syslog	It is possible to instruct NG/AMS to produce log entries into the UNIX syslog. This is only done when certain important events occur. Such events are error conditions, and handling of archive requests. Whether or not to log into the syslog is specified in the configuration file (CFG: "NgamsCfg.Log:SysLog"). It is possible to specify an "ID" which is written in each log entry in the syslog (CFG: "NgamsCfg.Log:SysLogPrefix"). This makes it possible to filter out log for a certain context at a later stage. An example of some syslog entries produced by NG/AMS is: … Feb 20 12:58:04 w2p2nbu python: DFSLog:2002-02-20T12:58:04.200 Error w2p2nbu NGAMS_ER_DISK_INACCESSIBLE:3004:ERROR: Disk with ID: Slot ID: 3 - Disk ID: IC35L080AVVA07-0-VNC400A4C1G8RA is not accessible (writable). Feb 20 12:58:04 w2p2nbu python: DFSLog:2002-02-20T12:58:04.410 Error w2p2nbu NGAMS_ER_DISK_INACCESSIBLE:3004:ERROR: Disk with ID: Slot ID: 4 - Disk ID: IC35L080AVVA07-0-VNC400A4G0KZ8A is not accessible (writable). Feb 21 23:43:56 w2p2nbu python: DFSLog:2002-02-21T23:43:56.800 Notice w2p2nbu Disk with ID: IC35L080AVVA07-0-VNC400A4C1607A - Name: LS-FitsStorage3-M-000027 - Slot No.: 5 - running low in available space (4938 MB)! Feb 22 09:29:40 w2p2nbu python: DFSLog:2002-02-22T09:29:40.740 Notice w2p2nbu Marked Main Disk with ID: IC35L080AVVA07-0-VNC400A4C1607A - Name: LS-FitsStorage3-M-000027 - Slot No.: 5 - as 'completed' - PLEASE CHANGE! Feb 22 09:29:40 w2p2nbu python: DFSLog:2002-02-22T09:29:40.770 Notice w2p2nbu Marked Replication Disk with ID: IC35L080AVVA07-0-VNC400A4C1622A - Name: LS-FitsStorage3-R-000027 - Slot No.: 6 - as 'completed' - PLEASE CHANGE! …
Verbose Log	The Verbose Logs are written to stdout. They contain more details information than the two other types of logs. This type of log is usually used for debugging, trouble shooting and test purposes. The Verbose Log Level is adjusted via a command line parameter (-v <level). If this parameter is not specified, no Verbose Log output is produced. The format of the Verbose Logs is as follows: <ISO 8601 time stamp>:<module>:<method>:<line no.>:<log type>> <log message> An example of Verbose Log output is: … 2002-02-25T13:48:52.340:ngamsServer.py:handleHttpRequest:538:INFO> Handling HTTP request: client_address=('134.171.2.3', 47718) - method=POST - path=|ARCHIVE| - content-disposition=file_uri="WFI.1999-09-04.fits"; wait="1" - content-type=ngas/archive-push - host=arcus1 - content-length=17573760 2002-02-25T13:48:52.430:ngamsCmdHandling.py:cmdHandler:667:INFO> Received command: ARCHIVE 2002-02-25T13:48:52.450:ngamsCmdHandling.py:handleCmdArchive:354:INFO> Handling Archive Push Request ... 2002-02-25T13:48:52.460:ngamsArchiveUtils.py:dataHandler:454:INFO> Archiving file: WFI.1999-09-04.fits with mime-type: image/x-fits ... 2002-02-25T13:49:13.320:ngamsFitsPlugIn.py:ngamsFitsPlugIn:122:INFO> Plug-In handling data for file with URI: WFI.1999-09-04.fits 2002-02-25T13:49:39.230:ngamsArchiveUtils.py:dataHandler:525:INFO> NGAMS_INFO_FILE_ARCHIVED:4020:INFO: Successfully archived file with URI: WFI.1999-09-04.fits. Time (s): 46.7714939117 2002-02-25T13:49:39.240:ngamsArchiveUtils.py:dataHandler:526:INFO> NGAMS_INFO_FILE_ARCHIVED:4020:INFO: Successfully archived file with URI: WFI.1999-09-04.fits …

The Log Level is a number in the range from 1 to 5, whereby 1 is the 'high-level' logs and 5 is the lowest (deepest) level, providing the most thorough information. The interpretation of the various Log Levels is as follows:

Level	Description
1	The lowest Log Level, which only provides a brief summary of the actions performed. Errors and warnings are always logged.
2	This level provides more thorough information of the actions performed.
3	This level performs a quite extensive set of logs describing in details the various actions carried out by NG/AMS and the plug-ins invoked by this. For logging in the log file, there should normally not be logged with a higher level than 3.

4	This level provides a very profound set of information. It is usually only used for debugging and test purposes and for locating errors occurring in the system.
5	The deepest level provides a quite extensive set of logs. Some of the log will be quite repetitive, and logs may be produced cyclically from e.g. the Janitor and the Data Consistency Check Services. The quantity of log information produced is quite big, and if logging into a log file with this level, care should be taken that it may grow in size very rapidly.

The level (intensity) with which there should be logged as well as name of Local Log File and a prefix for the syslog entries, can be specified in the NG/AMS Configuration (CFG: "NgamsCfg.Log"). For further information about this specific properties see chapter 4.

2.12 Email Notification

Apart from the various types of logging described in section 2.9, it is also possible to let NG/AMS notify various recipients about important events occurring via email.

The events that can trigger broadcasting of such event messages are:

Event	*Description*
Alert Notification	An Alert Message is generated as a result of a serious problem encountered. Such a problem may not be recoverable, and it is likely necessary to do some manually intervention. Normally preventative actions should be undertaken immediately.
Error Notification	An Error is the result of a problem encountered, which is not of a very serious character. Often an error situation is provoked by an external request, which is illegal for some reason. Could e.g. be that it is tried to archive a file when the system is in Offline State. Depending on the type of error intervention should be undertaken (ASAP).
Disk Space Notification	A Disk Space Notification is sent out when a certain threshold of minimum free disk capacity is reached. This message is meant only as a 'warning' that the Storage Set is about to be full. No actions are needed, apart from maybe verifying that sufficiently Storage Sets with free disk space are available.
Disk Change Notification	A Disk Change Notification is sent out to indicate that a Storage Set is full and should be removed from the archiving unit and normally replaced with a new disk set. See also section 2.15.
No Disk Space Notification	If no more free Storage Sets are available, a No Disk Space Notification Message is sent out to the subscribers of this event. Since this is a severe problem, a special Notification Message is dedicated for this problem.
Data Error Notification	If the Data Consistency Check Service encounters errors/problems with data files, a Data Error Notification Message is sent to the subscribers of this event. The files which were found to be 'bad' in some way should be analyzed to find out what is causing the problem. It could be caused by physical problems of the disk, or that due to long storage on the disk, failures start to occur. Problems with a 'problematic' file are normally only reported once. I.e., if the problem is not solved, there will be no more notification about the problematic file.

The Notification Setup is configured in the NG/AMS Configuration (CFG: "NgamsCfg.Notification"). For further information about this specific property see chapter 4. An example Disk Change Notification Message can be found in section 2.15.

2.13 Simulation Mode

It is possible to operate the NG/AMS Server in Simulation Mode, whereby a number of features are disabled or are working slightly different than in Normal Mode. One of the major differences is that it is possible to run without the availability of storage disks. Simulated storage disks are created as directories in the Mount Root Point. These are of the format: "<mount root point>/<storage set ID>-Main|Rep-<sim. slot ID>".

Another difference is that the Online and Offline Plug-Ins are not executed since no disks need to be mounted or unmounted. The disk information about the disks is generated/simulated and written in the DB.

For the clients of NG/AMS there is no visible difference between running in Normal Mode or in Simulation Mode. Also the internal aspects are the same, so that e.g. the DB is updated in the same manner in Simulation Mode as in Normal Mode. The Simulation Mode can be quite useful for developing and testing e.g. DHPIs and DPPIs.

It should be mentioned that it is possible to have a fully operational NG/AMS installation running in Simulation Mode on a 'normal' workstation archiving and retrieving data to/from one of the system disks of the workstation.

To enable/disable the Simulation Mode, the attribute "NgamsCfg.Ngams:Simulation" in the configuration is used. See also chapter 4.

2.14 Back-Log Buffering

Back-Log Buffering is used to temporarily buffer data, which for some reason, not necessarily related to the quality of the data, prevents NG/AMS from performing a proper archiving of the data file. An example of such an event, is e.g. if the DB connection is lost for a while.

As seen from figure 2.2, the Back-Log Buffer Area could be located in the NG/AMS Root Mount Directory as it is practical to collect the data of NG/AMS under a single point. If a problem occurs during the handling of an Archive Request, a file with a unique name will be created in this area and the data of the request stored in this file. The reply to the archive request will indicate the problem, i.e. that the data was Back-Log Buffered. No further actions are needed from the client that issued the archive request. The NG/AMS Server has an internal thread, Janitor Thread, which runs periodically and tries to clean up the NG/AMS environment. Furthermore it archives Back-Log Buffered data. If such an attempt fails due to one of the reasons justifying for Back-Log Buffering, the data will be kept in the Back-Log Buffer and a new attempt to archive it repeated later. If the attempt fails for another reason, the data will be moved to the Global Bad Files Area shown in figure 2.2. In this case a Notification Message will be sent out to the subscribers of Error Notification Messages, and the appropriate information logged in the log output targets specified.

In the NG/AMS Configuration it can be specified if Back-Log Buffering should be performed, as well as the parent directory for the Back-Log Buffer Directory. For further information about this specific property consult chapter 4.

2.15 Disk Space Monitoring

During the archiving process, NG/AMS monitors constantly the state of the set of disks currently installed. If the amount of data on a Storage Set reaches a certain limit defined by a configuration parameter, a Notification Message can be send out to a list of subscribers for this event. This event is a pre-warning that this Storage Set is going to be completed (full) within a limited time. The latter depends on the threshold defined in the configuration file. When a Storage Set considered as completed, another type of Notification Message can be broadcast to a number of subscribers. This message will indicate that the Storage Set was full and needs to be replaced. The appearance of such a message is as follows (example):

```
Subject: LSNau1: CHANGE DISKS
Date:    Fri, 25 Jan 2002 01:06:26 +0100 (MET)
From:    ngasmgr.w2p2nau@eso.org

Error Message:

PLEASE CHANGE DISKS:

Main Disk:
- Logical Name: LS-FitsStorage1-M-000024
- Slot ID:      5

Replication Disk:
- Logical Name: LS-FitsStorage1-R-000024
- Slot ID:      6
```

The Logical Name(s) (Disk Label(s)) as well as the Slot in which the disk(s) are hosted are indicated in the mail. When such a message is received by the NGAS responsible (operators) it is advisable to carry out the suggested changes as soon as possible to avoid saturation.

2.16 The NG/AMS Server Command Interface

The NG/AMS Server command interface is based on the standard HTTP protocol. This makes it easy to interface easily to the NG/AMS Server from a variety of different kinds of clients in a simple and straightforward manner. E.g. from a WEB browser (better if XML enabled) it is possible to query the status of an NG/AMS Server:

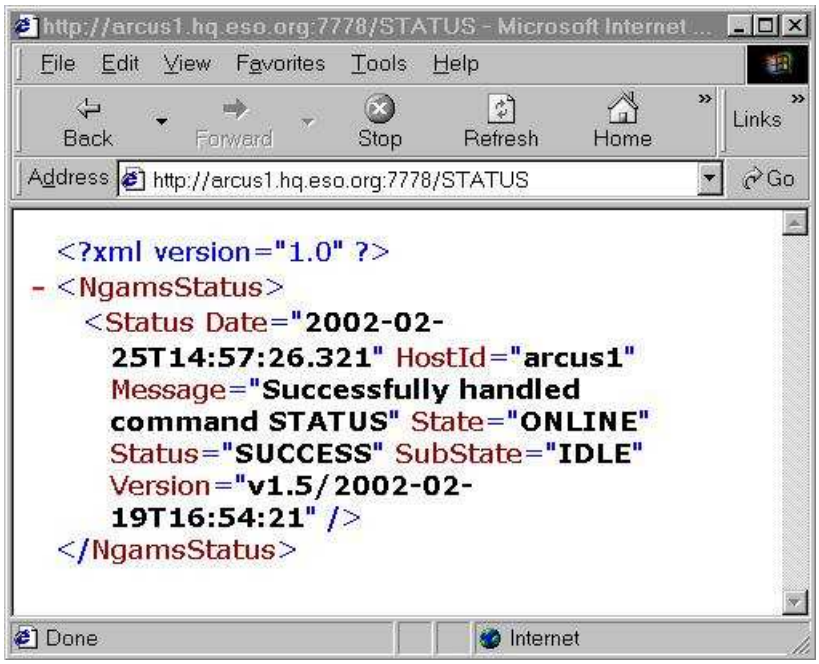


Figure 2.5: Interaction with an NG/AMS Server from WEB browser.

Also a standard utility like "telnet" can be used to interact with NG/AMS, e.g. to issue a command like OFFLINE:

```
arcus1 jknudstr:~ 34 > telnet arcus1 7778
Trying 134.171.2.3...
Connected to arcus1.hq.eso.org.
Escape character is '^]'.
GET OFFLINE

HTTP/1.0 200 OK

<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-02-25T15:26:09.164" HostId="arcus1" Message="Successfully handled command OFFLINE"
          State="OFFLINE" Status="SUCCESS" SubState="IDLE" Version="v1.5/2002-02-19T16:54:21"/>
</NgamsStatus>

Connection closed by foreign host.
arcus1 jknudstr:~ 35 >
```

In general, the NG/AMS Python or C based command interface tools, should be used when interacting with NG/AMS from the shell. See sections 3.2 and 3.3 for more information about these tools.

For more in-depth information about the NG/AMS command interface, consult the chapters 19 and 5.

2.17 Data Consistency Checking

The NG/AMS Server can be configured to carry out a periodic consistency check of the set of data files, which is stored on the disks installed in an NGAS system. The following checks are carried out:

- It is checked if files are registered in the DB but are not found on the disk.
- Checksum value for each file is checked according to the value registered in the NGAS DB for the file.
- It is checked if files are found in the Storage Area of the storage disks, which are not registered in the DB.

In case discrepancies are found in the data holding on the disks in connection with an NGAS host, a Data Inconsistency Notification Message is sent out. This has the format, e.g.:

```
Subject: NGAS-arcus2-7778: DATA INCONSISTENCY(IES) FOUND
Date:    Fri, 25 Jan 2002 01:06:26 +0100 (MET)
From:    jknudstr@eso.org

Error Message:

DATA INCONSISTENY(IES) FOUND IN DATA HOLDING:
Date:            2002-02-12T15:32:05.424
NGAS Host:       arcus2
Inconsistencies: 1

Problem Description                    File ID                          Version
-------------------------------------------------------------------------------
ERROR: Inconsistent checksum found     TEST.2001-05-08T15:25:00.123     3
-------------------------------------------------------------------------------
```

If files are found, which do not have the checksum properly set, NG/AMS will calculate the checksum using the DCPI specified in the configuration, and send a Data Inconsistency Notification to the subscribers of this type of message.

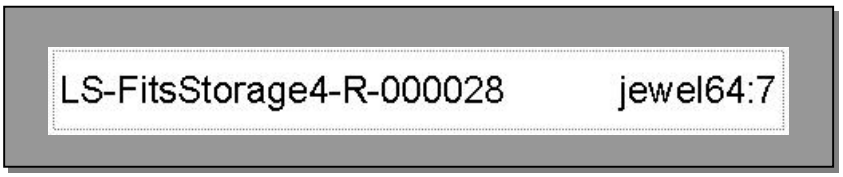
It is possible to enable and disable the Data Consistency Checking Service (CFG: "NgamsCfg.-FileHandling:DataCheckActive"). In addition it is possible to allocate a priority to the data checking thread to calibrate the CPU consumption (CFG: "NgamsCfg.FileHandling:DataCheckPrio"). It is also possible to specify how disks and files are checked, whereby this can either be done sequentially or randomly (CFG: "NgamsCfg.FileHandling:DataCheckDiskSeq", "NgamsCfg.FileHandling:DataCheckFileSeq"). A minimum cycle time for one iteration of the service can also be defined (CFG: "NgamsCfg.FileHandling:DataCheckMinCycle"). If the checking is carried out in less then the specified minimum cycle time, the service will be suspended for a while. A parameter is used to configure the service to produce a log entry after each iteration with summary information about the check carried out. This log entry has the following contents:

```
2002-02-26T02:52:00.640 [INFO] Number of files checked: 9529. Amount of data checked: 582478.078 MB.
Time for checking: 45139.280 s
```

The configuration parameters mentioned above are described in more detail in chapter 4.

2.18 Label Printing

A label to stick on the disk cases can be produced by NG/AMS by means of the LABEL command. The text on the label is the Logical Name allocated to the disk. In addition printed on the label is the host ID and the Slot ID. The exact appearance of a label is as follows (generated by the Brother P-Touch 9200 DX label printer):



The part with the Host ID + Slot ID should be removed from the label before sticking it on to the disk case.

The LABEL command takes as input the Slot ID in which the disk to generate the label for is installed. In addition thew Host ID of the host in which the disk is installed.

2.19 Security

It is important to keep in mind that the NG/AMS SW does not come with any security measures built-in when it comes to preventing undesirable intruders (hackers) from connecting to the server and invoking services. This must all be handled at the level of the operating system and network (firewals etc.). What is supported are checks to disable certain services and to ensure that only a limited set of plug-ins can be invoked by clients.

3 The NG/AMS Server and Utilities

Three 'executables' are provided within the NG/AMS package. These 1) The NG/AMS Server - "ngamsServer(.py)", 2) The NG/AMS Python Client - "ngamsPClient(.py)", and 3) The NG/AMS C Client - "ngamsCClient". These executables are described in the following sections.

3.1 NG/AMS Server Command Line Interface

By calling the NG/AMS Server without command line parameters (or illegal ones), the following online help is printed on stdout:

```
arcus1 jknudstr:~/dev/ngams/ngamsServer 44 > ngamsServer
Correct usage is:

ngamsServer -cfg <cfg file> [-v <level>] [-version]
            [-locLogFile <log file>] [-locLogLevel <level>
            [-sysLog <level>] [-sysLogPrefix <prefix>]
            [-force] [-autoOnline] [-d]

-cfg <cfg file>        NG/AMS Configuration File.

-v <level>             Verbose Mode + Level.

-version               Print out version of server.

-locLogFile <file>     Name of Local Log File.
-locLogLevel <level>   Level for logging in Local Log File.

-sysLog                Switch syslog logging on.
-sysLogPrefix <prefix> Prefix for syslog logging.

-d                     Debugging Mode.

-force                 Force execution even though PID File found.

-autoOnline            Bring the server to Online State
                       automatically after initialization.

Note: The values given on the command line, overwrites the
      ones given in the NG/AMS Configuration File.

(c) ESO/DMD 2001 - NGAS Project - http://archive.eso.org/NGAST

arcus1 jknudstr:~/dev/ngams/ngamsServer 45 >
```

3.2 Python Shell Utility

By invoking the NG/AMS Python command utility without input parameters (or with illegal ones), the following online help is written on stdout:

```
arcus1 jknudstr:~/dev/ngams/ngamsPClient 51 > python ngamsPClient.py

> ngamsPClient -host <host> -port <port> -cmd <command>
               [-v <level>] [-version>] [-d]
               [-noWait] [-status]
               [-fileUri <file URI> [-mimeType <mt>](*)
               [-fileId <file ID> [-outputFile <file | dir>] [-process <DPPI>]](***)

-host <host>:   Host where the NG/AMS Server is running.

-port <port>:   Port number used by the NG/AMS Server.

-cmd <command>: Command to issue: ARCHIVE(*), EXIT,
                INIT, LABEL(**), ONLINE, OFFLINE,
                RETRIEVE(***), STATUS

-v <level>:     Verbose output level.

-version:       Print version and exit.

-d:             Run in Debug Mode.

-noWait:        Don't wait for the NG/AMS Server
                to terminate the handling of the command.

-status:        Dump the status message sent by the NG/AMS
                Server to stdout.

*: For the ARCHIVE command the following parameter must be given:

-fileUri <uri>:  URI pointing to the file to be archived.

                   file:/home/data/Image1.fits

                 This will result in an Archive Pull
                 Request. It can also be given directly as
                 a filename, e.g.:

                   /home/data/Image1.fits

                 This will result in an Archive Push Request.

-mimeType <mt>:  If it is not possible to determine
                 the mime-type of a data file from
                 the filename, the mime-type must be
                 explicitly given with the ARCHIVE
                 command.

**: For the LABEL command the following parameter must be given:

-slotId <id>:     Slot ID for which to generate label.

***: For the RETRIEVE command the following parameters are defined:

-fileId <id>:     File ID of the file to retrieve, e.g.:

                    ISAAC.1999-04-11T05:03:21.035

-outputFile ...:  Directory or file in which to dump
                  a file when retrieving files.

-process ...:     Name of Data Processing Plug-In to
                  execute on the data before sending
                  it back to the requestor.

(c) ESO/DMD 2001 - NGAS Project - http://archive.eso.org/NGAST
```

3.3 C Shell Utility

By invoking the NG/AMS C based command utility without input parameters (or with illegal ones), the following online help is written on stdout[2]:

```
arcus1 jknudstr:~/dev/ngams/ngamsCClient 60 > ngamsCClient

> ngamsCClient -host <host> -port <port> -cmd <command>
               [-noWait] [-status]
               [-fileUri <URI>](*)
               [-fileId <id> [-process <dppi> ...]
                [-outputFile <file | dir>]](**)

-host  <host>:    Host where NG/AMS Server is running:

                    host.domain.org

-port  <port>:    Port number used by the NG/AMS Server.

-cmd <command>:   Command to issue: ARCHIVE(*), EXIT, LABEL,
                  ONLINE, OFFLINE, RETRIEVE(**), STATUS(***).

-noWait:          Don't wait for the NG/AMS Server to finish
                  the handling of the request.

-status:          Dump the status message sent by the NG/AMS
                  Server to stdout.

*: For the ARCHIVE command the following parameter must be given:

-fileUri <uri>:   URI pointing to the file to be archived.
                  Can be given as e.g.:

                    file:/home/data/Image1.fits

                  This will result in an Archive Pull
                  Request. It can also be given directly as
                  a filename, e.g.:

                    /home/data/Image1.fits

                  This will result in an Archive Push Request.

**: For the RETRIEVE command the following parameters are defined:

-fileId <id>:     File ID of the file to retrieve, e.g.:

                    ISAAC.1999-04-11T05:03:21.035

-fileVersion <#>: Version of file to retrieve.

-process ...:     Name of DPPI to invoke on the data.

-outputFile ...:  Directory or file in which to dump a file
                  when retrieving files.

***: For the LABEL command the following parameter is defined:

-slotId <ID>:     Slot ID to print label for.

(c) ESO/DMD 2001 - NGAS Project - http://archive.eso.org/NGAST
```

[2] It is the intention to keep the command line interfaces of the NG/AMS Python and C Clients identical. Minor changes in the present interfaces will be fixed. Both interface are shown here for completeness (although they are in principal identical).

4 *EXPERT:* Configuring NG/AMS

The NG/AMS SW is implemented in a very flexible way to be able to adjust the system for various scenarios. In order to obtain this, a number of parameters can be adjusted in the NG/AMS Configuration.

This chapter contains a description of these parameters. The format for the NG/AMS configuration also includes the Header Element which is a generic standard header for XML documents. This header is not described here. An example of an NG/AMS Configuration can be found in section 4.3. The DTD defining the format of the NG/AMS Configuration can be found in section 4.1.

Element	*Element*	*(Element:)Attribute*	*Description*
Ngams[3]			The Ngams Element contains all the overall parameters defining an NG/AMS installation.
		ArchiveName	The name of the Archive to which this NG/AMS installation belongs. Could be e.g. "ESO".
		GroupId	The Group ID for the NGAS installation. Should describe to which sub-group of NGAS hosts a particular host belongs. This is used when building the Logical Name for the storage media.
		PortNo	The port number used by the NG/AMS Server for serving HTTP requests.
		OnlinePlugIn	Name of the Online Plug-In that is invoked by NG/AMS when going Online. The purpose of this plug-in is to prepare the system for operation and to extract and return the information about the storage disks available on the system.
		OnlinePlugInPars	This parameter makes it possible to transfer special configuration information to an Online Plug-In to make it possible to re-use it in various contexts. The format of this is not strictly defined, but it is suggested to use a format as: "<par>=<value>,<par>=<value>…".
		OfflinePlugIn	Offline Plug-In invoked by NG/AMS while the system is being brought to Offline State. This plug-in prepares the system for standby.
		OfflinePlugInPars	Context specific parameters for the Offline Plug-In, See also "OnlinePlugInPars".
		LabelPrinterPlugIn	The Label Printer Plug-In, is used to control the label printer, which is used to generate labels to stick on to the disk cases. This is invoked by NG/AMS when executing the LABEL command.
		LabelPrinterPlugInPars	Context specific parameters for the Label Printer Plug-In. See also "OnlinePlugInPars".
		Replication	Used to enable/disable the file replication service of NG/AMS. A value of 1 indicates that NG/AMS should perform replication. Replication is disabled when set to 0.
		BlockSize	The size (in bytes) applied by NG/AMS when sending and receiving packages via network. This can be used to tune the data rates for obtaining optimal transfer times.
		Simulation	Used to enable/disable the Simulation Mode of NG/AMS. A value of 1 sets NG/AMS in Simulation Mode, a value of 0 makes NG/AMS operate in Normal Mode.
		MountRootDirectory	Complete path for the directory where the Data Disks are mounted. In this directory also the Global Bad Files Directory, the Staging Area and Processing Area + other internal files, are located. See also 0.
		CentralUnit	Used to configure a system as a 'Central Unit'. A Central Unit 'takes posses' of the disks under its control. It sets the DB column "ngas_disks.protected" to 1 to indicate that this entry should not be overwritten.
		AllowArchiveReq	Used to enable/disable the ability of an NG/AMS installation to handle Archive Requests. A value of 1 enables handling of Archive Requests, a value of 0 disables it. If an Archive Request is issued when disabled, the NG/AMS Server will return an error reply to the request.
		AllowRetrieveReq	Used to enable/disable the ability of an NG/AMS installation to handle Retrieve Requests. A value of 1 enables handling of Retrieve Requests, a value of 0 disables it. If a Retrieve Request is issued when disabled, the NG/AMS Server will return an error reply to the request.
		AllowProcessingReq	Used to enable/disable the ability of an NG/AMS installation to handle Processing Requests. A value of 1 enables handling of Processing Requests, a value of 0 disables it. If a Processing Request is issued when disabled, the NG/AMS Server will return an error reply to the request.

[3] The Ngams element is a sub-element of the NgamsCfg element. Latter is not shown to avoid one level in the hierarchy in the table shown above.

		ForceProxyMode	This flag is used to indicate to an NG/AMS installation that it should always try to act as proxy when handling a Retrieve Request of data which is located on remote location. I.e., if enabled (=1), NG/AMS will request the data file from the remote NGAS host, and send this back to the requestor.
		JanitorSuspendTime	Time (in seconds) the Janitor Thread should be disabled after each iteration.
		BackLogBuffering	Used to enable/disable Back-Log Buffering. See also 2.14.
		BackLogBufferDirectory	Name of directory in which the Back-Log Buffer will be located. Must be given as the complete path.
Db			Element that contains the definition of the DB connection.
		Server	DB server name.
		Name	Name of DB within the DB server to use.
		User	The name of the DB user.
		Password	Password for the DB user.
MimeTypes			Define the mime-types that are known to an NG/AMS installation, and which can be handled by this installation.
	MimeTypeMap		Defines the mapping between a mime-type and the extension of the file.
		MimeType	Defines the name of the mime-type for this type of data.
		Extension	Defines the extension used by data files of the given mime-type.
StorageSet			Defines one Storage Set of an NGAS host. This can consist of one or two disks.
		StorageSetId	The ID used to refer to the Storage Set.
		MainDiskSlotId	The Slot ID allocated to the Main Disk of the Storage Set.
		RepDiskSlotId	The Slot ID allocated to the Replication Disk of the Storage Set. If no Replication Disk is specified, no replication will be performed. NG/AMS will also not update the DB with the information about the Replication File. I.e., it is possible to operate the system completely without the concept of a back-up file -- with the corresponding risks!
		Mutex	used to indicate to NG/AMS that it should provide mutual access for writing on the given Storage Set. This may increase the performance for the writing of individual files, and may prevent too much de-fragmentation of the data on the disk. It may however, decrease the overall rate for writing on the disks.
FileHandling			Defines various properties in connection with the data/file handling.
		GlobalBadDirLoc	Name of directory in which the Global Bad File Directory is located. Must be given as the complete path.
		ProcessingDirectory	Name of directory in which the Processing Directory is located. Must be given as the complete path.
		PathPrefix	Prefix that will be added to the Mount Root Point, under which the actual Storage Area (Storage Directories) of the data disk are located.
		ChecksumPlugIn	Name of plug-in which is invoked by NG/AMS to calculate the checksum value for a file. See also chapter 15.
		ChecksumPlugInPars	Context specific parameters for the Checksum Plug-In. See also "OnlinePlugInPars".
		DataCheckActive	Flag to enable/disable the Data Consistency Check. A value of 1 means enabled, a value of 0 disabled. See also section 2.17.
		DataCheckPrio	The priority of the Data Consistency Checking Thread. The priority is a number greater then or equal 0, which is used to adjust the time the thread is suspended. The lower the priority, the less time the thread is suspended, i.e., the more CPU it will consume.
		DataCheckMinCycle	The minimum cycle time for carrying out a Data Consistency Check. This is used to avoid to stress the data disks too much, since checking involves heavy disk I/O. The format is ISO-8601 like: "<days>T<hours>:<minutes>:<seconds>". For instance: "2T03:30:00" means that the Data Consistency Check is at most, executed once per 2 days, 3 hours and 30 minutes. If the check takes up shorter time the thread will be suspended up till that time. Otherwise, if the check takes up longer time, the thread would start the next iteration immediately after terminating the previous.
		DataCheckDiskSeq	Used to indicate if the Data Consistency Check should be carried out sequentially or randomly of the disk. The valid values are: "SEQUENTIAL" and "RANDOM". If disks are checked sequentially, NG/AMS will check the disks sorted according to Slot ID. Otherwise disks will be checked randomly.
		DataCheckFileSeq	Used to indicate if the Data Consistency Check should be carried out sequentially or randomly of the files stored on a disk. The valid values are: "SEQUENTIAL" and "RANDOM". If files are checked sequentially, NG/AMS will check the files of a disk sorted according to ingestion date. Otherwise the files will be checked randomly.

		DataCheckLogSummary	Used to enable/disable logging of a report log entry after one Data Consistency Checking cycle. A value of 1 means enabled, otherwise 0 must be specified.
Stream			The Stream Element is used to define the properties for one Stream of data. A Stream is dealing with one specific type of data. See also section 2.7.
		MimeType	Mime-type of the data for this Stream. A corresponding Mime-Type Mapping definition for this mime-type must be defined in the MimeTypes Element.
		PlugIn	Name of the DHPI to invoke to handle data with this mime-type. See also section 2.7 and chapter 13.
		PlugInPars	Context specific parameters to hand over to the DHPI.
	StorageSetRef		Used to refer to the Storage Set, on to which this kind of data should be stored.
		StorageSetId	ID of the Storage Set, on to which data of this type can be stored.
Monitor			Contains information about properties to monitor within an NG/AMS installation.
		MinFreeSpaceWarningMb	The limit (in MBs) for the minimum amount of free disk space for when an Email Disk Space Warning Notification Message should be send to the subscribers of this event. See also description of Notification Element and section 2.12.
		FreeSpaceDiskChangeMb	The limit (in MBs) for the minimum amount of free disk space for when an Email Change Disk Notification Message should be send to the subscribers of this event. See also description of Notification Element and section 2.12.
Log			The Log Element defines properties for the logging performed by an NG/AMS installation.
		SysLog	Used to enable/disable production of log entries into the UNIX syslog. A value of 1 means enabled and 0 disabled.
		SysLogPrefix	A prefix to add in the Syslog entries. This makes it possible to filter out logs generated by NG/AMS. See also section 2.11 about logging.
		LocalLogFile	Complete filename of the Local Log File into which NG/AMS can produce log entries.
		LocalLogLevel	The Log Level for producing logs in the UNIX Syslog log file. Should be a value in the range [0; 5]. A value of 5 means the highest intensity.
Notification			The Notification Element is used to set up the Notification Service. See also section 2.12.
		SmptHost	The name of the mail server used for handling the distribution of Notification Emails. Must be given as e.g.: "smtphost.hq.eso.org".
		Sender	Used to set the sender of the Notification Message to make it clear for the subscribers from where/by whom the email was sent.
		Active	Used to globally enable/disable the Email Notification Service. A value of 1 means enabled and 0 means disabled.
	AlertNotification ErrorNotification DiskSpaceNotification DiskChangeNotification NoDiskSpaceNotification DataErrorNotification		For each type of Email Notification an element is used to defined the list of subscribers. These are contained within the elements listed to the left. Several subscribers can be defined for each type of Notification Message.
		EmailRecipient:Address	Email address of a subscriber of a given type of Notification Message.

4.1 *EXPERT:* NG/AMS Configuration DTD - "ngamsCfg.dtd"

The DTD for the NG/AMS configuration is based on the "ngamsInternal.dtd" (see section 4.2), which defines the NG/AMS specific elements used in the NG/AMS configuration. The contents is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % XmlStd SYSTEM "XmlStd.dtd">
%XmlStd;
<!ENTITY % NgamsInternal SYSTEM "ngamsInternal.dtd">
%NgamsInternal;

<!--
  E.S.O.

  "@(#) $Id: ngamsCfg.dtd,v 1.2 2002/02/26 16:24:44 safcvs Exp $"

  Who        When        What
  ********   **********  ************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  ngamsCfgNau.dtd defines the contents and lay-out of the
  configuration file loaded by the NG/AMS Server at start-up.

  Consult the DTD ngamsInternal.dtd for further information. It
```

```
  contains the actual definition of the elements of the NG/AMS
  Configuration.
  -->

<!--
  The NgamsCfg element is the root element of the NG/AMS
  Configuration for NG/AMS NAU Systems.
  -->
<!ELEMENT NgamsCfg (Header,
                    (Ngams, Db, MimeTypes, StorageSet*,
                     Stream*, FileHandling, Monitor, Log, Notification?))>

<!-- oOo -->
```

4.2 *EXPERT:* NG/AMS Base DTD - "ngamsInternal.dtd"

The base DTD is used to define various XML components (elements), which can be re-used in various deducted DTD/XML documents. The contents is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  E.S.O.

  Who        When        What
  ********   **********  ************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  The ngamsInternal.dtd defines various common elements to be used
  in the NG/AMS XML documents.
  -->


<!--
  The Ngams Element defines properties for the individual
  installation of NG/AMS.

  Attributes:
    ArchiveName:            Name of the archive - e.g. ESO-VLT.

    GroupId:                ID that indicates to which group of NGAS Systems
                            this NGAS System belongs. The Group ID is used to
                            generate the Logical Disk Name.

    PortNo:                 Port number to use for the NG/AMS HTTP server.

    OnlinePlugIn:           Plug-In utility invoked by NG/AMS when
                            going Online to prepare the system and to
                            obtain the information about the current
                            disk configuration and status of the disks.

    OnlinePlugInPars:       Input parameters to the Online Plug-In.

    OfflinePlugIn:          Plug-In utility invoked by NG/AMS when
                            going Offline to prepare the system for
                            standby mode.

    OfflinePlugInPars:      Input parameters to the Online Plug-In.

    Replication:            Indicates if file replication should be
                            carried out by this NG/AMS (0|1).

    BlockSize:              Block size applied when receiving and
                            sending data via HTTP (bytes).

    Simulation:             Simulation system "1" otherwise "0".

    MountRootDirectory:     Base directory used as root directory when
                            mounting the disks.

    CentralUnit:            Indicates if this is a central unit or not.
                            If a unit is configured as a Central Unit,
                            it takes posess of the disks and ensures
                            that their entries are not overwritten
```

```
                           in the DB (0|1).

    AllowArchiveReq:       Allow clients to issue Archive Requests
                           to the system (0|1).

    AllowRetrieveReq:      Allow clients to retrieve data from this
                           NG/AMS installation (0|1).

    AllowProcessingReq:    Allow clients to perform processing requests
                           on this NG/AMS installation (0|1).

    ForceProxyMode:        If a server which is contacted with a Retrieve
                           Request, finds a file to be retrieved, on
                           another NGAS host, it will act as proxy and
                           send back the file to the requestor if
                           possible.

    JanitorSuspendTime:    Period of time the Janitor Thread is suspended
                           after each iteration. Should be given as
                          '<days>T<hours>:<minutes>:<seconds>'.

    BackLogBuffering:      Enable/disable Back-Log Data Buffering (0|1).

    BackLogBufferDirectory: Directory that will host the "back-log" directory
                           where data files are buffered for later handling
  -->
<!ELEMENT Ngams EMPTY>
<!ATTLIST Ngams ArchiveName            CDATA          #REQUIRED
                GroupId                CDATA          #REQUIRED
                PortNo                 CDATA          #REQUIRED
                OnlinePlugIn           CDATA          #REQUIRED
                OnlinePlugInPars       CDATA          #REQUIRED
                OfflinePlugIn          CDATA          #REQUIRED
                OfflinePlugInPars      CDATA          #REQUIRED
                LabelPrinterPlugIn     CDATA          #IMPLIED
                LabelPrinterPlugInPars CDATA          #IMPLIED
                Replication            CDATA          #IMPLIED
                BlockSize              CDATA          #REQUIRED
                Simulation             (0|1)          "0"
                MountRootDirectory     CDATA          #REQUIRED
                CentralUnit            (0|1)          #REQUIRED
                AllowArchiveReq        (0|1)          #REQUIRED
                AllowRetrieveReq       (0|1)          #REQUIRED
                AllowProcessingReq     (0|1)          #REQUIRED
                ForceProxyMode         (0|1)          #IMPLIED
                JanitorSuspendTime     CDATA          #REQUIRED
                BackLogBuffering       (0|1)          #IMPLIED
                BackLogBufferDirectory CDATA          #IMPLIED>


<!--
  The Db element defines properties for the interaction
  with the NGAS DB.

  Attributes:
    Server:    Name of DB server.

    Name:      Name of the DB to use.

    User:      The DB user name to connect as.

    Password:  The password for the DB user.
  -->
<!ELEMENT Db EMPTY>
<!ATTLIST Db Server       CDATA  #REQUIRED
             Name         CDATA  #REQUIRED
             User         CDATA  #REQUIRED
             Password     CDATA  #REQUIRED>


<!--
  The MimeTypes Element contains a mapping between the mime-types used
  by NG/AMS and the extension names of the data files.

  The element MimeTypeMap contains the mapping between each mime-type
  and the corresponding extension.
```

```
  Attributes:
    MimeType:      Mime-type, e.g. "ngas/fits".

    Extension:     Extension of data file, e.g. "fits" (without
                   leading dot).
  -->
<!ELEMENT MimeTypes (MimeTypeMap+)>
<!ELEMENT MimeTypeMap EMPTY>
<!ATTLIST MimeTypeMap   MimeType   CDATA  #REQUIRED
                        Extension  CDATA  #REQUIRED>


<!--
  The StorageSet Element defines properties for each 'storage unit',
  consisting of two disks (Main Disk and Replication Disk).

  Attributes:
    StorageSetId:       ID used to refer to this particular
                        storage unit (string).

    MainDiskSlotId:     Slot ID for Main Disk (string).

    RepDiskSlotId:      Slot ID for Replication Disk (string).

    Mutex:              Indicates if mutual exclusion access should be
                        enforced to the disks. If this is enabled, only
                        one data reception session will write data to
                        that slot (disk), and not simultaneous as otherwise
                        done if several files for the same disk is received
                        at the same time. 1 = mutual exclusion (integer).
  -->
<!ELEMENT StorageSet EMPTY>
<!ATTLIST StorageSet StorageSetId        CDATA  #REQUIRED
                     MainDiskSlotId      CDATA  #REQUIRED
                     RepDiskSlotId       CDATA  #REQUIRED
                     Mutex               (0|1)  "0">


<!--
  The FileHandling element defines properties for the file handling.

  Attributes:
    GlobalBadDirLoc:      Storage location where files determined as bad
                          will be stored. Under the path name given a
                          directory "bad-files" will be created.

    ProcessingDirectory:  Base directory in which temporary files
                          will be stored during processing.

    PathPrefix:           A prefix which will be used when building up the
                          target filename. Could e.g. be used as to build
                          a filename like:

                          <mount point>/<PathPrefix>/<add. path>/<filename>

                          - but it is up to the Data Handling Plug-In
                         to use it in a way appropriate for the context.

   ChecksumPlugIn:        DCPI - Data Checksum Plug-In, which generates the
                          checksum of a data file.

   ChecksumPlugInPars:    Input parameters to the DCPI.

   DataCheckActive:       Enable/disable Data Check Service (1 = activate).

   DataCheckPrio:         Priority of Data Check Thread (1 = highest priority).
                          with this parameter it is possible to adjust the
                          amount of CPU power the Data Check Thread should
                          consume. The higher the number, the lesser CPU
                          the check will consume. At the same time, it
                          will take more time to carry out the check of
                          the data holding.

   DataCheckMinCycle:     Indicates the minimum allowed time for carrying out
                          one check cycle (ddThh:mm:ss). If this is specified
```

```
                        e.g. to 24 hours, and one check cycle would only
                        take 11 hours, the check would be suspended for
                        13 hours before running again.

    DataCheckDiskSeq:   Used to indicate if disks should be checked
                        sequentially (ordered according to the Slot
                        ID), or randomly (SEQUENTIAL|RANDOM).

    DataCheckFileSeq:   Used to indicate if files on a disk should be checked
                        sequentially (ordered according to the ingestion
                        date), or randomly (SEQUENTIAL|RANDOM).

    DataCheckLogSummary: If set to 1, a summary log info will be generated
                        each time a complete check of the data holding
                        of one NGAS host has been carried out.
  -->
<!ELEMENT FileHandling EMPTY>
<!ATTLIST FileHandling GlobalBadDirLoc      CDATA               #IMPLIED
                       ProcessingDirectory  CDATA               #REQUIRED
                       PathPrefix           CDATA               #REQUIRED
                       ChecksumPlugIn       CDATA               #IMPLIED
                       ChecksumPlugInPars   CDATA               #IMPLIED
                       DataCheckActive      (0|1)               #IMPLIED
                       DataCheckPrio        CDATA               #IMPLIED
                       DataCheckMinCycle    CDATA               #IMPLIED
                       DataCheckDiskSeq     (SEQUENTIAL|RANDOM) #IMPLIED
                       DataCheckFileSeq     (SEQUENTIAL|RANDOM) #IMPLIED
                       DataCheckLogSummary  (0|1)               #IMPLIED>

<!--
  The Stream Element defines properties for the handling of data
  streams into the NGAS System.

  The StorageSetRef contains an attribute with a reference to a StorageSet.

  Attributes:
  MimeType:      Mime-type identifying this type of data.

  PlugIn:        A plug-in command that will be executed to 1) Check the
                 consistency of the data file of the given mime-type,
                 2) Generate the necessary information for the NGAS DB.

  PlugInPars:    Parameters which will be transferred to the plug-in
                 function.
  -->
<!ELEMENT Stream (StorageSetRef+)>
<!ATTLIST Stream MimeType       CDATA  #REQUIRED
                 PlugIn         CDATA  #REQUIRED
                 PlugInPars     CDATA  #IMPLIED>
<!ELEMENT StorageSetRef EMPTY>
<!ATTLIST StorageSetRef  StorageSetId CDATA  #REQUIRED>


<!--
  The Monitor element defines properties for ingestion/handling
  of data files.

  Attributes:
    MinFreeSpaceWarning:    Indicates the free disk space limit before issuing
                            an Warning Log Message (MB).

    FreeSpaceDiskChangeMb:  Indicates the free disk space limit before
                            changing disk (MB).
  -->
<!ELEMENT Monitor EMPTY>
<!ATTLIST Monitor MinFreeSpaceWarningMb  CDATA #REQUIRED
                  FreeSpaceDiskChangeMb  CDATA #REQUIRED>


<!--
  The Log Element defines properties for the logging performed by
  the NG/AMS Server.

  Attributes:
    SysLog:         Switch on UNIX syslog logging (0|1).
```

```
    SysLogPrefix:   Prefix (tag) written first in the syslog entries
                    (in the data part).

    LocLogFile:     Indicates a name of a local log file. Should be
                    complete path.

    LocalLogLevel:  Log level for producing logs into the local
                    log file ([0; 5]).
  -->
<!ELEMENT Log EMPTY>
<!ATTLIST Log SysLog          (0|1)         #REQUIRED
              SysLogPrefix    CDATA         #REQUIRED
              LocalLogFile    CDATA         #REQUIRED
              LocalLogLevel   (0|1|2|3|4|5) #REQUIRED>


<!--
  The Notification Element is used to define subscribers (e-mail
  recipients) that will receive an e-mail when certain events
  occur. For the moment the following events can trigger e-mails
  to one or more recipients:

    o Alert Logs.
    o Error Logs.
    o Disk Running Full (Min. Free Disk Space).
    o Disk Change (Disk Change Log).
    o No more disk space.
  -->
<!ELEMENT Notification (AlertNotification?, ErrorNotification?,
                        DiskSpaceNotification?, DiskChangeNotification?,
                        NoDiskSpaceNotification?,
                        DataErrorNotification?)>
<!ATTLIST Notification  SmptHost   CDATA  #REQUIRED
                        Sender     CDATA  #REQUIRED
                        Active     (0|1)  #REQUIRED>
<!ELEMENT EmailRecipient EMPTY>
<!ATTLIST EmailRecipient Address   CDATA  #REQUIRED>
<!ELEMENT AlertNotification (EmailRecipient+)>
<!ELEMENT ErrorNotification (EmailRecipient+)>
<!ELEMENT DiskSpaceNotification (EmailRecipient+)>
<!ELEMENT DiskChangeNotification (EmailRecipient+)>
<!ELEMENT NoDiskSpaceNotification (EmailRecipient+)>
<!ELEMENT DataErrorNotification (EmailRecipient+)>

<!-- oOo -->
```

4.3 *EXPERT:* NG/AMS Configuration - Example

In the following, an example NG/AMS Configuration is listed:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE NgamsCfg SYSTEM "ngamsCfg.dtd">

<!--
  E.S.O.

  Who        When        What
  ********   **********  ************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  This is a template/example NG/AMS Configuration for an NG/AMS NAU System.

  Consult the DTDs ngamsCfg.dtd and ngamsInternal.dtd for further
  information.
  -->

<NgamsCfg>

  <Header Name="ngamsCfgNau.xml"
          Type="NGAMS-CONFIGURATION-NAU"
          Context="NGAMS"
          Release="1.0"
          Source="jknudstr@eso.org"
```

```
        Revision="@(#) $Id: ngamsCfgNau.xml,v 1.42 2002/02/26 11:03:37 safcvs Exp $">
    <Description>
    This XML document contains information for a template NGAMS
    configuration file.
    </Description>
  </Header>

  <Ngams  ArchiveName="ESO-ARCHIVE"
          GroupId="JKN"
          PortNo="7778"

          OnlinePlugIn="ngamsDiskInfo_3ware_Escalade_6800"
          OnlinePlugInPars="http://angast1.hq.eso.org:1080/technical.html"

          OfflinePlugIn="ngamsDummyPlugIn"
          OfflinePlugInPars=""

          LabelPrinterPlugIn="ngamsBrotherPT9200DxPlugIn"

LabelPrinterPlugInPars="dev=/dev/null,font_file=/home/jknudstr/saf/ngams/ngamsData/ngamsBrotherPT9200Dx
Fonts.fnt"

          Replication="1"

          BlockSize="8192"

          Simulation="1"

          MountRootDirectory="/home/jknudstr/ngams"

          CentralUnit="0"

          AllowArchiveReq="1"
          AllowRetrieveReq="1"
          AllowProcessingReq="1"

          ForceProxyMode="1"

          JanitorSuspendTime="0T00:03:00"

          BackLogBuffering="1"
          BackLogBufferDirectory="/home/jknudstr/ngams"/>

  <Db     Server="LUXSRV"
          Name="ngas"
          User="ngas"
          Password="ngas_pw"/>

  <MimeTypes>
    <MimeTypeMap  MimeType="image/x-fits"         Extension="fits"/>
    <MimeTypeMap  MimeType="ngas/fits-hdr"        Extension="hdr"/>
    <MimeTypeMap  MimeType="ngas/log"             Extension="log"/>
    <MimeTypeMap  MimeType="ngas/nglog"           Extension="nglog"/>
    <MimeTypeMap  MimeType="application/x-gfits"  Extension="fits.gz"/>
    <MimeTypeMap  MimeType="application/x-cfits"  Extension="fits.Z"/>
  </MimeTypes>

  <StorageSet StorageSetId="FitsStorage1"
              MainDiskSlotId="1"
              RepDiskSlotId="2"
              Mutex="1"/>

  <StorageSet StorageSetId="FitsStorage2"
              MainDiskSlotId="3"
              RepDiskSlotId="4"
              Mutex="1"/>

  <StorageSet StorageSetId="PafStorage"
              MainDiskSlotId="5"
              RepDiskSlotId="6"
              Mutex="0"/>

  <StorageSet StorageSetId="LogStorage"
              MainDiskSlotId="7"
              RepDiskSlotId="8"
              Mutex="0"/>
```

```
  <Stream  MimeType="image/x-fits"
           PlugIn="ngamsFitsPlugIn"
           PlugInPars="compression=compress -f,checksum_util=utilFitsChecksum,
                       checksum_result=0/0000000000000000">
    <StorageSetRef StorageSetId="FitsStorage1"/>
    <StorageSetRef StorageSetId="FitsStorage2"/>
  </Stream>

  <Stream  MimeType="ngas/log"
           PlugIn="ngamsLogPlugIn">
    <StorageSetRef StorageSetId="LogStorage"/>
  </Stream>

  <Stream  MimeType="ngas/nglog"
           PlugIn="ngamsNgLogPlugIn"
           PlugInPars="">
    <StorageSetRef StorageSetId="LogStorage"/>
  </Stream>

  <FileHandling  GlobalBadDirLoc="/home/jknudstr/ngams"
                 ProcessingDirectory="/home/jknudstr/ngams"
                 PathPrefix="saf"
                ChecksumPlugIn="ngamsGenCrc32"
                ChecksumPlugInPars=""
                DataCheckActive="0"
                DataCheckPrio="25"
                DataCheckMinCycle="0T00:03:00"
                DataCheckDiskSeq="SEQUENTIAL"
                DataCheckFileSeq="SEQUENTIAL"
                DataCheckLogSummary="1"/>

  <Monitor MinFreeSpaceWarningMb="0"
           FreeSpaceDiskChangeMb="20"/>

  <Log     SysLog="1"
           SysLogPrefix="DFSLog"
           LocalLogFile="/home/jknudstr/ngams/log/LogFile.nglog"
           LocalLogLevel="3"/>

  <Notification SmptHost="smtphost.hq.eso.org"
                Sender="jknudstr@eso.org"
                Active="0">
    <AlertNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </AlertNotification>

    <ErrorNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </ErrorNotification>

    <DiskSpaceNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </DiskSpaceNotification>

    <DiskChangeNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </DiskChangeNotification>

    <NoDiskSpaceNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </NoDiskSpaceNotification>

    <DataErrorNotification>
      <EmailRecipient Address="jknudstr@eso.org"/>
    </DataErrorNotification>

  </Notification>

</NgamsCfg>

<!-- oOo -->
```

5 *EXPERT:* NG/AMS Server Communication Protocol

The NG/AMS command interface is based on the HTTP protocol, which is a widely used standard protocol. This makes it easy to interface various client applications with NG/AMS. In this chapter the details of the NG/AMS command interface are described.

Using the NG/AMS Python and C APIs, the client application does not have to worry about the format of the replies generated by NG/AMS. It is therefore recommended whenever possible to use the APIs provided with the NG/AMS package.

5.1 *EXPERT:* Format of NG/AMS HTTP Command Messages

The format of the NG/AMS messages is defined as follows:

Archive Push Request:

```
POST /ARCHIVE HTTP/1.0[1]
User-Agent: <user agent>
Content-Type: ngas/archive-push
Content-Length: <length>
Content-Disposition: file_uri=<uri>[;wait=1|0]

<data>
```

Example:

```
POST /ARCHIVE HTTP/1.0
User-Agent: NG/AMS C-API
Content-Type: ngas/archive-push
Content-Length: 69120
Content-Disposition: file_uri="/tmp/TestFile.fits";wait="1"

~  ‡v}yzy~ƒ}{uˆ~,‰…tcv,, …
```

Archive Pull Request + Other Commands:

```
GET <command>?[<parameter>=<value>] HTTP/1.0
User-Agent: <user agent>
```

Example, Archive Pull Request:

```
GET ARCHIVE?file_uri="file:///tmp/SmallFile.fits"&wait="1" HTTP/1.0
User-Agent: NG/AMS C-API
```

The exact list of parameters for each command are described in chapter 19.

5.2 *EXPERT:* Format of the NG/AMS HTTP Reply

The format of replies from NG/AMS is defined as follows:

```
HTTP/<HTTP version> <HTTP response code> <message>
Server: <server ID>
Date: <date for generating reply>
Expires: <expiration date (= Date:)>
Content-Type: <mime-type>
Content-Length: <data length>

<data>
```

An example of a reply to an Archive Request is:

```
HTTP/1.0 200 OK
Server: v1.5/2002-02-19T16:54:21
Date: Tue, 26 Feb 2002 13:58:00 GMT
Expires: Tue, 26 Feb 2002 13:58:00 GMT
Content-Type: text/xml
Content-Length: 1171

<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-02-26T14:06:54.670" HostId="acngast1" Message="Successfully handled Archive Push Request
          for data file with URI: SmallFile.fits" State="ONLINE" Status="SUCCESS" SubState="IDLE"
          Version="v1.5/2002-02-19T16:54:21"/>
  <DiskStatus Archive="ESO-ARCHIVE" AssociatedDiskId="IBM-DTLA-305040-YJ0YJ075523" AvailableMb="39050"
              BytesStored="182225988" Checksum="" Completed="0" CompletionDate=""
              DiskId="IBM-DTLA-305040-YJ0YJ070913" HostId="acngast1"
              InstallationDate="2002-02-19T13:13:03.000" LogicalName="LS-FitsStorage4-M-000001"
              Manufacturer="IBM" MountPoint="/NGAS/data7" Mounted="1" NumberOfFiles="685" Protected="0"
              SlotId="7" TotalDiskWriteTime="1230.90544678" Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1246906309" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="TEST.2001-05-08T15:25:00.123"
                FileName = "saf/2001-05-08/626/TEST.2001-05-08T15:25:00.123.fits.Z" FileSize="53546"
                FileStatus="00000000" FileVersion="626" Format="application/x-cfits" Ignore="0"
                IngestionDate="2002-02-26T14:06:54.000" UncompressedFileSize="69120"/>
  </DiskStatus>
</NgamsStatus>
```

In a reply to a Retrieve Request the data returned will be contained in the message rather than the NG/AMS Status XML document shown above. Such a reply thus looks like this, e.g.:

```
HTTP/1.0 200 OK
Server: v1.5/2002-02-19T16:54:21
Date: Tue, 26 Feb 2002 14:14:43 GMT
Expires: Tue, 26 Feb 2002 14:14:43 GMT
Content-Type: application/x-cfits
Content-Length: 53546
Content-Disposition: attachment; filename="TEST.2001-05-08T15:25:00.123.fits.Z"

<data>
```

It is foreseen at a later stage to make it possible to query several files simulataneously with one query. This means that the mime-type "multipart/mixed" will be used as the overall mime-type of the reply and that each part has its proper mime-type defined.

5.3 *EXPERT:* Format of the NG/AMS Redirection HTTP Response

If an NG/AMS Server is not configured to always act as a proxy when data is being requested by a client, HTTP redirection response messages may be generated and send back to the requestor. The format of such redirection responses is:

```
HTTP/1.0 303 Method
Server: <server ID>
Date: <date>
Expires: <date>
Location: <URL pointing to actual location of file>
Content-Type: text/xml
Content-Length: <length>

<NG/AMS status document>
```

An example of such a redirection HTTP response is:

```
HTTP/1.0 303 Method
Server: v1.5/2002-02-12T10:52:10
Date: Tue,  Jan 01:34:40 2 GMT
Expires: Tue,  Jan 01:34:40 2 GMT
Location: http://jewel64:7777/RETRIEVE?file_id="WFI.2001-09-25T21:19:17.508"
Content-Type: text/xml
Content-Length: 339

<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2001-01-02T01:34:40.656" HostId="jewel68" Message="NGAMS_INFO_RETRIEVE_REDIRECT:4024:INFO:
          Redirection URL: http://jewel64:7777/RETRIEVE?file_id = WFI.2001-09-25T21:19:17.508"
          State="ONLINE" Status="SUCCESS" SubState="BUSY" Version="v1.5/2002-02-12T10:52:10"/>
</NgamsStatus>
```

The client must then re-issue the Retrieve Request to the alternative location given in the redirection response and will be able to get access to the data directly from that location (if the system permits). It should be mentioned that it is normally more efficient to request the data directly from the location where it is actually located rather than using NG/AMS as a proxy server. Again, using the NG/AMS APIs this is all handled transparently for the client application.

6 *EXPERT:* The NGAS DB

The NG/AMS SW is based on three tables in the NGAS DB. These are:

- **ngas_hosts**: Contains information about the hosts in an NGAS installation.
- **ngas_disks**: Contains information about the disks, which have been registered in an NGAS installation.
- **ngas_files**: Contains information about each files, which have been archived in an NGAS system.

The present release of NG/AMS is based on Sybase ASE, but it is foreseen to be able to make the SW work with other DBMS'.

The system is capable of operating without the availability of the "ngas_hosts" table. In addition it is possible to operate with a DB where the "ngas_hosts" is available but is empty. This however only, under certain conditions. It is however recommended to create and to populate this table under normal circumstances when operating an NGAS cluster.

Usually it is not foreseen that external applications perform queries directly into the NGAS DB. I.e., all information needed, should be retrieved via the NG/AMS Server. Apart from saving external applications from knowing technical details about the NGAS DB, this has the advantage of making such external applications independent of the DBMS used by an NG/AMS installation. For this reason, it is not guaranteed that 100% backwards compatibility is maintained when it comes to the format of the NGAS DB.

In the following sections the exact contents of the NGAS tables is described.

6.1 *EXPERT:* Table - "ngas_hosts"

Column	*Type*	*Value*	*Description*
db_id	numeric(8,0)	identity	Internal data base ID. This is generated automatically.
host_id	varchar(32)	not null	ID of the NGAS host, e.g. "jewel65". Should be given only as the name, i.e., without the domain name.
domain	varchar(30)	not null	Domain name of the NGAS host, e.g. "hq.eso.org".
ngas_port	int	null	The port number used by the NG/AMS Server when waiting for HTTP requests.
ip_address	varchar(20)	not null	The IP address of the NGAS host.
ngas_type	varchar(10)	not null	The type of NGAS host, i.e., which role it has. The value of this is not used by the NG/AMS SW. Suggested values could be "NAU" - NGAS Archiving Unit, "NBU" - NGAS Buffering Unit, "NCU" - NGAS Central Unit, "NMU" - NGAS Master Unit.
mac_address	varchar(20)	null	The MAC address coded into the network card used for communication by the NGAS host. This is an address like: "05:4E:14:8A:11:2B".
ngas_retrieve	tinyint	null	Indicates if this NGAS (NG/AMS) is configured to accept Retrieve Requests
n_slots	tinyint	null	Number of slots in the NGAS system.
ngas_version	varchar(20)	null	Version of the NG/AMS running on the NGAS host.
cluster_name	varchar(10)	null	Name of the NGAS cluster this system belongs to.
installation_date	datetime	null	Date the OS and NG/AMS running on the NGAS host have been installed.

6.2 *EXPERT:* Table - "ngas_disks"

Column	*Type*	*Value*	*Description*
disk_id	varchar(128)	not null	The ID of the disk. This information is extracted by the Online Plug-In from the BIOS of the disk drive. This is the unique identifier of the disk.
archive	varchar(64)	not null	The name of the archive to which the disk belongs. The value for this is taken from the NG/AMS Configuration.

installation_date	datetime	not null	The date for registering the disk the first time. Subsequent re-registering do not change the value of this column.
type	varchar(64)	not null	Describe the type of the media, e.g.: "MAGNETIC DISK/ATA". The value for this is generated by the Online Plug-In
manufacturer	varchar(64)	null	The manufacturer of the disk. Could e.g. be "IBM" or "Seagate". This value is generated by Online Plug-In.
protected	tinyint	null	This flag is set to 1 when the disk is registered in an NGAS system, which is configured as 'Central System' in its configuration. In the NGAS installation of ESO, this is used to indicate that the DB replication from the observatory sites to the central archive site, should be disabled for entries with this column set to 1.
logical_name	varchar(128)	not null	The Logical Name of the disk, is a 'human readable' (unique) ID for the disk. It is generated by NG/AMS when the disk is registered the first time.
associated_disk_id	varchar(128)	not null	The Associated Disk ID is the ID of the disk used together with a disk in a Storage Set. This column is only of relevance when a disk is used in an Archiving NGAS System, in a Storage Set together with another disk. Otherwise, the files within an NGAS system, should be considered file-wise and each disk as an independent storage unit.
host_id	varchar(32)	null	The ID of the host where a disk is currently registered. If the disk is not registered in any NGAS host, this will be set to "".
slot_id	varchar(32)	null	The ID of the slot in the NGAS host, in which the disk is currently registered. If the disk is not registered, this will be "".
mounted	tinyint	null	Used to indicate if a disk is mounted or not (1 = mounted, 0 - not mounted).
mount_point	varchar(128)	null	Used to give the (complete) name of the mount point where the disk is mounted. If the disk is not mounted this will be "".
number_of_files	int	not null	Indicates how many data files that have been archived on the disk.
available_mb	int	not null	Used to indicate the amount of available storage capacity still free on the disk (given in MBs).
bytes_stored	numeric(20, 0)	not null	Used to indicate the amount of data stored on the disk (given in bytes).
completed	tinyint	not null	Used to indicate that the disk is 'completed', i.e., NG/AMS has been archiving files on the disk, and has reached the threshold specified in the configuration file.
completion_date	datetime	null	Set by NG/AMS when the disk reached the threshold for completion.
checksum	varchar(64)	null	The global checksum value for the disk. ***Note, this is not set for the moment!***
total_disk_write_time	float	null	Total time spent on writing data on the disk.

6.3 *EXPERT:* Table - "ngas_files"

Column	*Type*	*Value*	*Description*
disk_id	varchar(128)	not null	ID of the disk where the file is stored.
file_name	varchar(255)	not null	Name of the file. This must be given relative to the mount point of the disk.
file_id	varchar(64)	not null	File ID allocated to the file by the DHPI. The set of File ID, Disk ID and File Version, uniquely defines a file.
file_version	int	default 1	Version of the file. The first version is number 1.

format	varchar(32)	not null	Format of the file. This is generated by the DHPI. Should normally be the mime-type of the file, as stored on the disk.
file_size	numeric(20, 0)	not null	Size of the file. This must be given in bytes. If the file is compressed, the compressed file size must be given as value for this column.
uncompressed_file_size	numeric(20, 0)	not null	If the file was compressed this indicates the size of the uncompressed file. If the file is not compressed this will be equal to the file_size.
compression	varchar(32)	null	The compression method applied on the file. Could be e.g. "gzip". This should indicate clearly how the file has been compressed, to make it possible to decompress it at a later stage.
ingestion_date	datetime	not null	Date the file was ingested/archived.
ignore	tinyint	null	Used to indicate that this file should be ignored (1 = ignore). If set to one, this entry for this file, will not be taken into account by NG/AMS when files or information about files is queried.
checksum	varchar(64)	null	Checksum of the file. This value is generated by the checksum plug-in specified in the configuration.
checksum_plugin	varchar(64)	null	Name of the checksum plug-in used to generate the checksum for the file. This is used by NG/AMS when performing the Data Consistency Checking of data files. NG/AMS in this way, invokes the same plug-in as was used to generate the checksum originally.

file_status	char(8)	default '00000000'	Current status of the file. The status should be seen as a sequence of bytes, each with a certain signification what concerns the condition and status of the file. These bytes are used to indicate the following (when set to 1). The bytes in the status are counted from left to right:
			
			`1:  The File Checksum is incorrect.`
			`2:  File being checked.`
			`3-8:  Not used.`
			
			The bytes 3-8 may be used at a later stage.

7 *EXPERT:* The C-API

Together with the NG/AMS package an API to be used for interfacing C applications with the NG/AMS Server is provided. This is provided in the form of a small library with a set of functions making it easy to communicate from client applications to the NG/AMS Server. Also a number of various macros are provided by the C-API.

The sources and the header file for the C-API is contained in the module: "ngams/ngamsCClient". This contains the files:

File	*Description*
Makefile	Make that generates/compiles the C-API. Can be invoked with the parameters clean and all, e.g.: "make clean all".
ngams.h	Header file for the NG/AMS C-API module. This contains the definition of the function prototypes, and the definition of various macros that can be used in the clients built using the NG/AMS C-API.
ngamsCClient.C.c	The source file for the NG/AMS C based command line utility. See also section 3.3 for more information about this tool.
ngamsCClientLib.c	The source file for the library functions provided by the NG/AMS C-API.

Compiling the "ngamsCClient" module, the following binaries are generated:

Binary	*Description*
libngams.a	The library to be linked with application using the NG/AMS C-API.
ngamsCClient	The binary/executable utility, which can be used to communicate with the NG/AMS Server from the command line. Refer to section 3.3 for further infomation.

In the following sections the header file for the NG/AMS C-API is listed. In addition the man-page for the C-API library functions is shown.

7.1 *EXPERT:* NG/AMS C-API - Header File: ngams.h

The source of the NG/AMS C-API header file can be found in the NG/AMS module as follows: "ngams/ngamsCClient/ngams.h". It contains the prototype definitions for the barious functions provided by the API, and also the definition of various macros.

7.2 *EXPERT:* NG/AMS C-API - Man Page

The man-page for the NG/AMS C-API contains the following information:

```
NAME
ngamsArchive(), ngamsOnline(), ngamsOffline(), ngamsExit(),
ngamsRetrieve2File(), ngamsStatus(), ngamsStat2Str(), ngamsCmd2Str(),
ngamsCmd2No() - C functions to interface to NG/AMS


SYNOPSIS
#include "ngams.h"

In general for the NG/AMS interface functions listed below,
the "host" parameter is the name of the host where the NG/AMS Server
is running. E.g.: "arcus2.hq.eso.org". The "port" parameter is the
socket port, which the NG/AMS Server is waiting on.

The parameter "noWait" is used to request an immediate reply to the
request, i.e. before the request has been handled.

The parameter "status" is a structure containing the following members:

Data Type       Member     Description
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ngamsSMALL_BUF  date       Date for handling query.
int             errorCode  Error code giving status for the query.
                           See #1.
ngamsSMALL_BUF  hostId     Host ID for host where the NG/AMS Server
                           is running.
```

```
ngamsHUGE_BUF   message    Message from the NG/AMS Server.
ngamsSMALL_BUF  status     Status of query ("OK" | "FAILURE").
ngamsSMALL_BUF  state      State of the NG/AMS Server.
ngamsSMALL_BUF  subState   Sub-State of the NG/AMS Server.
ngamsSMALL_BUF  version    Version of the NG/AMS Server.


#1: The following error codes (internal to the NG/AMS C API)
    are defined (data type: ngamsSTAT):

Error Macro             Description
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ngamsSTAT_SUCCESS       Query successfully executed.

ngamsERR_HOST           No such host.
ngamsERR_SOCK           Cannot create socket.
ngamsERR_CON            Cannot connect to host/server.

ngamsERR_WR_HD          Write error on socket while writing header.
ngamsERR_WR_DATA        Write error on socket while writing data.
ngamsERR_RD_DATA        Read error while reading data.
ngamsERR_INV_REPLY      Invalid answer from data server.

ngamsERR_FILE           Invalid filename specified.
ngamsERR_ALLOC_MEM      Cannot allocate memory.

ngamsERR_UNKNOWN_STAT   Unknown status code.
ngamsERR_UNKNOWN_CMD    Unknown command issued.
ngamsERR_INV_TARG_FILE  Invalid target filename specified.
ngamsERR_INV_PARS       Invalid parameters given.

ngamsSRV_OK             Request sucessfully handled by server.
ngamsSRV_REDIRECT       The reply is an HTTP redirection reponse.

ngamsSRV_INV_QUERY      Invalid query.


Apart from that, the errors defined by NG/AMS can be returned.


All functions return ngamsSTAT_SUCCESS in case of success. In case of
an error a termination status within the set of status codes
given above.


The following macros are defined for referring to NG/AMS commands:

Command Macros                            Description
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ngamsCMD_ARCHIVE/ngamsCMD_ARCHIVE_STR     Archive file.
ngamsCMD_EXIT/ngamsCMD_EXIT_STR           Make NG/AMS Server exit.
ngamsCMD_LABEL/ngamsCMD_LABEL_STR         Make NG/AMS print out a
                                          disk label.
ngamsCMD_ONLINE/ngamsCMD_ONLINE_STR       Bring NG/AMS Server Online.
ngamsCMD_OFFLINE/ngamsCMD_OFFLINE_STR     Bring NG/AMS Server Offline.
ngamsCMD_RETRIEVE/ngamsCMD_RETRIEVE_STR   Retrieve a file.
ngamsCMD_STATUS/ngamsCMD_STATUS_STR       Query status of NG/AMS.


The following functions are provided for interacting with NG/AMS:


ngamsSTAT ngamsArchive(const char          *host,
                       const int           port,
                       const char          *fileUri,
                       const char          *mimeType,
                       const int           noWait,
                       ngamsSTATUS         *status)
Archive a file in the NGAS system.

mimeType:    The mime-type of the file to archive. In some cases
             it is not possible for NG/AMS to determine the mime-type
             of a data file to be archived, e.g. when the file being is
             archived is RETRIEVEd from another NGAS host. For efficiency
             it is thus better to indicate the mime-type to enable
             NG/AMS to store the file directly on the target disk.
```

```
            If not use this can be put to "".


ngamsSTAT ngamsLabel(const char          *host,
                     const int           port,
                     const char          *slotId,
                     ngamsSTATUS         *status)
Send a LABEL command to the NG/AMS Server.


ngamsSTAT ngamsOnline(const char          *host,
                      const int           port,
                      const int           noWait,
                      ngamsSTATUS         *status)
Send an ONLINE command to the NG/AMS Server to bring it to Online State.


ngamsSTAT ngamsOffline(const char          *host,
                       const int           port,
                       const int           noWait,
                       ngamsSTATUS         *status)
Send an OFFLINE command to the NG/AMS Server to bring it to Offline State.


ngamsSTAT ngamsExit(const char          *host,
                    const int           port,
                    const int           noWait,
                    ngamsSTATUS         *status)
Send an EXIT command to the NG/AMS Server to make it
clean up and terminate execution.


ngamsSTAT ngamsRetrieve2File(const char          *host,
                             const int           port,
                             const char          *fileId,
                             const int           fileVersion,
                             const char          *processing[],
                             const char          *targetFile,
                             ngamsSTATUS         *status);
Send a RETRIEVE command to the NG/AMS Server to retrieve a
data file, and store this in a file on the local disk.

fileId:      ID of the file to retrieve.

fileVersion: Specific version of file to retrieve. If set to -1 the
             latest version will be retrieved.

processing:  Array with names of Data Processing DPPIs to apply on the
             data being retrieved. Must be terminated with NULL.
             If not used "{NULL}" can be given in, other it must be
             defined as, e.g.:

               char *processing[] = {"DPPI1", "DPPI2", "DPPI3", NULL};

targetFile:  If a valid filename is specified the data retrieved
             will be stored in a file with that name. If a directory
             is given, the data file retrieved will be stored in that
             directory with the name under which it is stored in NGAS.
             If this parameter is an empty string, it will be tried
             to stored the file retrieved under the NGAS archive name
             in the current working directory.


ngamsSTAT ngamsStatus(const char          *host,
                      const int           port,
                      ngamsSTATUS         *status)
Send a STATUS command to NG/AMS to query the current status of the
system. No parameters are defined at present.


ngamsSTAT ngamsStat2Str(const ngamsSTAT  statNo,
                        ngamsMED_BUF     statStr)
Convert a status code (ngamsSTAT) to a readable string.


ngamsSTAT ngamsCmd2Str(const ngamsCMD    cmdCode,
```

```
                        ngamsSMALL_BUF    cmdStr)
Convert an NG/AMS command given as a code (integer) to a string.


ngamsSTAT ngamsCmd2No(const ngamsSMALL_BUF    cmdStr,
                      ngamsCMD                *cmdCode)
Convert a command given as string into the corresponding code (integer).


char *ngamsEncodeUrlVal(const char *urlVal,
                         const int  skipScheme)
Encode the value given as input parameter to replace special
characters to make the value suitable for usage in a URL.

urlVal:            Value to be encoded.

skipScheme:        If the value is initiated with an HTTP scheme
                   (ftp:, http:, file:), this will not be encoded
                   if this flag is set to 1.


CAUTIONS
This is a first implementation of the module. Changes may be
introduced in order to improve the usability of the API.


EXAMPLES
To archive a file using the API the following must be called from
the application:

#include "ngams.h"

ngamsSTATUS      status;
if (ngamsArchive("wfinau", "7171", "/home/data/MyFile.fits", "",
                 0, &status) != ngamsSTAT_SUCCESS)
    {
    ngamsDumpErrStdout(&status);
    ... error handling ...
    }


To retrieve a file into the directory "/home/data/target_dir". The
name will be the same as the File ID:

#include "ngams.h"

ngamsSTATUS      status;
if (ngamsRetrieve2File("wfinau", "7171", "WFI.2001-10-21T23:24:03.925",
                       -1, {NULL}, "/home/data/target_dir",
                       &status) != ngamsSTAT_SUCCESS)
    {
    ngamsDumpErrStdout(&status);
    ... error handling ...
    }




- - - - - -
Last change:  26/02/02-16:45
```

8 *EXPERT:* The Python API

The NG/AMS Python API can be used by Python applications to interface with the NG/AMS Server in an easy and straightforward manner. The API hides most of the technical details of the NG/AMS communication interface.

To use the Python API, the following "import" statements must be contained in the client application:

```
from   ngams import *
import ngamsPClient

…
```

The API provides a class "ngamsPClient", which is contained in the module "ngamsPClient.py". The complete documentation for the API is contained as in-line, Python documentation strings in the source file. It is suggested to browse this documentation online using a WEB browser and the "pydoc" utility. See also section 17.1. Here only a summary of the API is given. The most significant methods provided by this class are:

Method	*Description*
archive()	Archive a file to NG/AMS. This can be either done by the Archive Pull Technique or by the Archive Push Technique. The technique applied depends on the File URI given.
exit()	Used to issue an EXIT command to a running NG/AMS Server.
getHost()/setHost()	Get/set the NGAS host reference indicating where the NG/AMS Server with which there is communicated is running.
getPort()/setPort()	Get/set the NGAS port reference indicating the port used by the NG/AMS Server with which there is communicated.
handleCmd()	Execute a command from a set of command line parameters; mostly relevant for the NG/AMS Python based command line utility based on the Python API.
init()	Send an INIT command to the NG/AMS Server.
label()	Send a LABEL command to the NG/AMS Server.
offline()	Send a OFFLINE command to the NG/AMS Server.
online()	Send a ONLINE command to the NG/AMS Server.
pushFile()	Carry out the necessary actions for performing an Archive Push Request.
retrieve2File()	Retrieve a file into a file on the local disk.
sendCmd()/sendCmdGen()	Handle the sending of a command and reception of the reply to this command.
status()	Send a STATUS command to the NG/AMS Server.

A small example application based on the NG/AMS Python API is listed in the following. It is used to archive a file:

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsPClientEx.py,v 1.1 2002/02/26 16:24:44 safcvs Exp $"
#
# Who       When        What
# --------  ----------  ----------------------------------------------------------
# jknudstr  26/02/2002  Created
#
"""
Small example application archiving a file.
"""

import sys
from   ngams import *
import ngamsPClient

# Check the input parameters.
if (len(sys.argv) != 4):
    print "Correct usage is:\n"
    print "ngamsPClientEx <host> <port> <file URI>\n"
    sys.exit(1)

# Get the parameters for handling the archiving.
host    = sys.argv[1]
port    = sys.argv[2]
fileUri = sys.argv[3]

# Create instance of NG/AMS Python API.
client = ngamsPClient.ngamsPClient(host, port)

# Execute the command.
status = client.archive(fileUri)

# Handle result - here we simply print the XML status message to stdout.
print status.genXml(0, 1, 1, 1).toprettyxml('  ', '\n')[0:-1]

#
# ___oOo___
```

This small test program will generate an output as the following on stdout while archiving the file (example):

```
arcus1 jknudstr:~/dev/ngams/ngamsPClient 92 > python ngamsPClientEx.py acngast1 7777
~/work/TEST_FILES/SmallFile.fits
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-02-26T16:34:36.184" HostId="acngast1" Message="Successfully handled Archive Push Request
          for data file with URI: SmallFile.fits" State="ONLINE" Status="SUCCESS" SubState="IDLE"
          Version="v1.5/2002-02-19T16:54:21"/>
  <DiskStatus Archive="ESO-ARCHIVE" AssociatedDiskId="IBM-DTLA-305040-YJ0YJ075523" AvailableMb="39050"
              BytesStored="182333080" Checksum="" Completed="0" CompletionDate=""
              DiskId="IBM-DTLA-305040-YJ0YJ070913" HostId="acngast1"
              InstallationDate="2002-02-19T13:13:03.000" LogicalName="LS-FitsStorage4-M-000001"
              Manufacturer="IBM" MountPoint="/NGAS/data7" Mounted="1" NumberOfFiles="687" Protected="0"
              SlotId="7" TotalDiskWriteTime="1230.94028784" Type="MAGNETIC DISK/ATA">
    <FileStatus Checksum="1246906309" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="TEST.2001-05-08T15:25:00.123"
                FileName="saf/2001-05-08/628/TEST.2001-05-08T15:25:00.123.fits.Z" FileSize="53546"
                FileStatus="00000000" FileVersion="628" Format="application/x-cfits" Ignore="0"
                IngestionDate="2002-02-26T16:34:35.000" UncompressedFileSize="69120"/>
  </DiskStatus>
```

9 *EXPERT:* The NG/AMS Plug-In API

The NG/AMS Plug-In API provides convenience functions to facilitate the implementation of the various types of plug-ins used within the context of NG/AMS. The actual thorough documentation is contained as inline Python documentation strings in the code itself. For further information about this issue consult section 17.1.

It is recommended to restrict the usage of functions from NG/AMS modules to only the one ones contained in the NG/AMS Plug-In API (Python Module: "ngamsPlugInApi.py"). It should be mentioned, that for the moment the amount of convenience functions provided is limited. Basically only the functions needed for implementing the plug-ins provided so far, have been considered in this context. If new functions are needed requests for these could be issued to the authors of NG/AMS. Here is an overview of the convenience functions provided by the NG/AMS Plug-In API for the moment:

Function	*Description*
determineMimeType()	From the filename of a data file, determine the mime-type of the file.
execCmd()	Execute a command on the shell, and return a tuple with the following information: [<exit code>, <stdout>, <stderr>].
genDhpiSuccessStat()	Generate a return status object as it must be returned from a DHPI.
genFileInfo()	Extract the information about a file and return this to the plug-in.
getFileSize()	Get the size of a file.
getFitsKeys()	Extract a number of FITS keyword cards from a FITS file.
notify()	Send a Notification Message from the plug-in.
parseDhpiPlugInPars()	Get the plug-in parameters. This function is dedicated to be used by DHPIs.
parseRawPlugInPars()	Parse the plug-in parameters. These are supposed to be defined in the following format: "<parameter>=<value>,<parameter>=<value>,…". The parameters and values are returned in a dictionary whereby the keys of the dictionary are the parameter names.
prepProcFile()	The function is used to create a copy of a file to be processed in the Processing Directory.

Examples of plug-in can be found in the chapters: 10 (System Online Plug-In), 11 (System Offline Plug-In), 12 (The Label Printer Plug-In), **13 (The Data Handling Plug-In)** , 14 (The Data Processing Plug-In) and 15 (The Data Checksum Plug-In).

Apart from the functions contained in the module "ngamsPlugInApi.py", the following classes are used for implementing the plug-ins: "ngamsServer", "ngamsConfig", "ngamsReqProps", "ngamsDppiStatus", "ngamsDb", "ngamsPhysDiskInfo" (NG/AMS Disk Dictionary). These classes are all described in more details in chapter 17.

To be able to write efficiently, plug-ins for NG/AMS, it is required to have a more a less profound overview of the NG/AMS SW, or at least this will be of major advantage, depending on the complexity of the tasks performed by the plug-ins. An overview of the NG/AMS SW is given in chapter 17.

10 *EXPERT:* The System Online Plug-In

The purpose of the System Online Plug-In, is to prepare the system for the Online State, where it must be fully operational according to the configuration. During this phase the storage disks are usually mounted and possibly checked for proper functioning and accessibility. A very essential task of a System Online Plug-In is to generate the so-called Disk Dictionary. This contains the 'physical' information about the disks installed in an NGAS system.

The plug-in is invoked by NG/AMS when it is going Online, i.e., either when it has received an ONLINE command or when it has been started with the "-autoOnline" command line parameter. The actual implementation depends highly on the the context (HW) in use and other specific requirements in connection with an NGAS system.

10.1 *EXPERT:* Interface of a System Online Plug-In

The System Online Plug-In must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A System Online Plug-In has an interface as shown in figure 12.1.

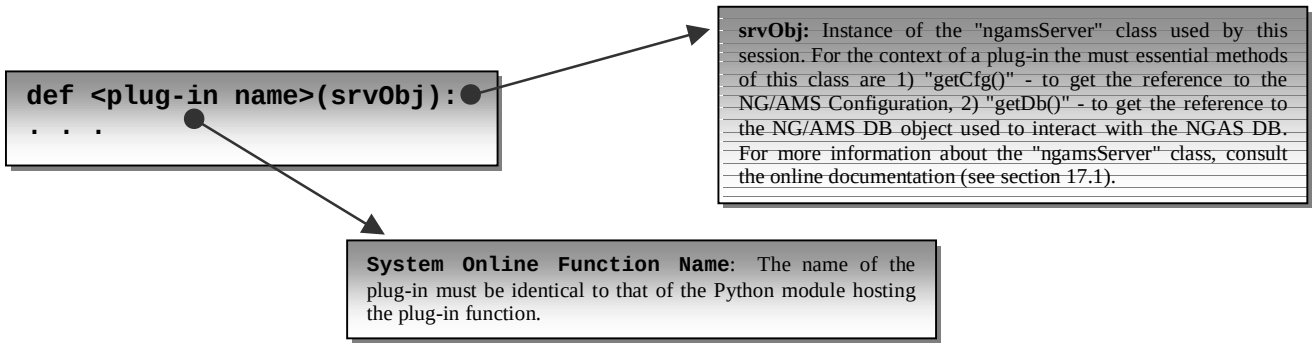


Figure 12.1: Function interface of a System Online Plug-In.

The return value of a System Online Plug-In is the NG/AMS Disk Dictionary. This must be generated by the plug-in. It is a standard Python dictionary with "ngamsPhysDiskInfo" objects stored in it. The Slot IDs of the disks are used as keys in the dictionary. The Disk Dictionary is very essential for the proper operation of NG/AMS. It is therefore crucial that the plug-in extracts and generates this information correctly for NG/AMS. The contents of the Disk Dictionary is illustrated in figure 12.2.

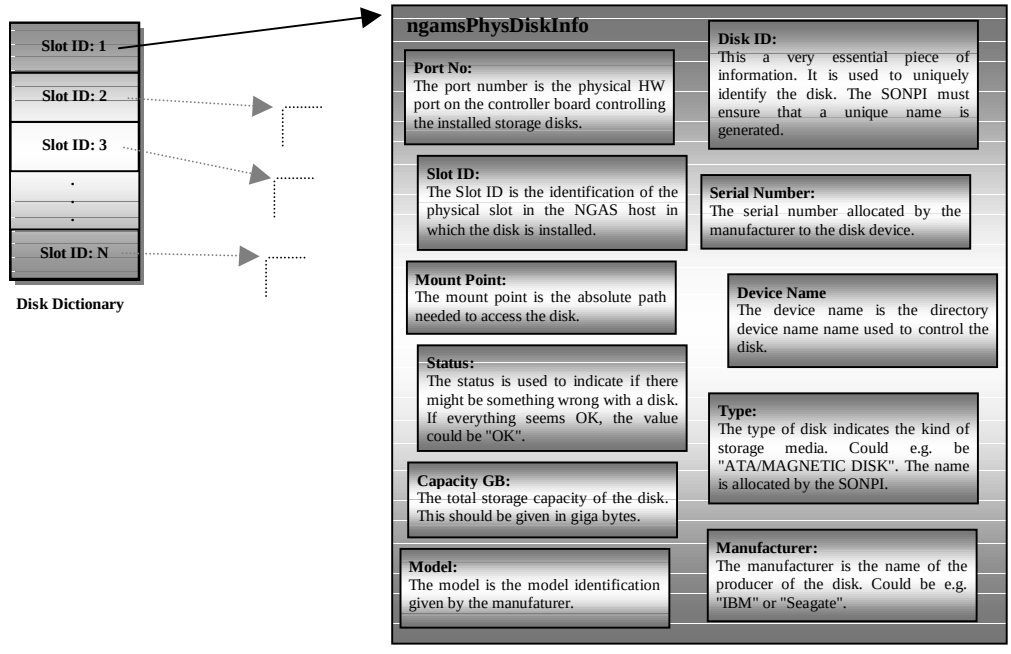


Figure 12.2: The NG/AMS Disk Dictionary.

An exception must be thrown in case errors occur during the process of bringing the system to Online State.

10.2 *EXPERT:* Overall Structure & Algorithm of a System Online Plug-In

The overall structure of a System Online Plug-In Python source is very similar to that of the DHPI. See 13.2 for further information about this issue.

10.3 *EXPERT:* Example System Online Plug-In

In the following an example System Online Plug-In, which is used for the moment for the NGAS installation for WFI at the La Silla 2.2m telescope. It is perhaps not a very good example of such a plug-in since most of the code is distributed in other modules. Please check the modules "ngamsEscalada6800Utils.py" and "ngamsLinuxSystemPlugInApi.py" for further information.

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsLinuxOnlinePlugIn.py,v 1.13 2002/02/27 17:18:26 safcvs Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  10/05/2001  Created.
#
"""
Module that contains System Online Plug-In.
"""
from   ngams import *
import ngamsPlugInApi
import ngamsLinuxSystemPlugInApi, ngamsEscalada6800Utils

def ngamsLinuxOnlinePlugIn(srvObj):
    """
    Function mounts all NGAMS disks and loads the kernel module for
    the IDE controller card. It returns the NGAMS specific
    disk info dictionary.

    srvObj:        Reference to instance of the NG/AMS Server
                   class (ngamsServer).

    Returns:       Disk info dictionary (dictionary).
    """
    rootMtPr = srvObj.getCfg().getMountRootDirectory()
    parDic = ngamsPlugInApi.\
             parseRawPlugInPars(srvObj.getCfg().getOnlinePlugInPars())
    stat = ngamsLinuxSystemPlugInApi.insMod(parDic["module"])
    if (stat == 0):
        msg = "Kernel module " + parDic["module"] + " loaded"
        info(1, msg)
        diskDic = ngamsEscalada6800Utils.parseHtmlInfo(parDic["uri"], rootMtPr)
        ngamsLinuxSystemPlugInApi.removeFstabEntries(diskDic)
        ngamsLinuxSystemPlugInApi.ngamsMount(srvObj.getDb(), diskDic)
        return diskDic
    else:
        errMsg = "Problem executing ngamsLinuxOnlinePlugIn"
        errMsg = genLog("NGAMS_ER_ONLINE_PLUGIN", [errMsg])
        error(errMsg)
        raise exceptions.Exception, errMsg

if __name__ == '__main__':
    """
    Main function.
    """
    import sys
    import ngamsConfig, ngamsDb

    setLogCond(0, "", 0, "", 1)

    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% python ngamsLinuxOnlinePlugIn <NGAMS cfg>\n"
        sys.exit(0)

    ngamsCfgObj = ngamsConfig.ngamsConfig()
    ngamsCfgObj.load(sys.argv[1])
    dbConObj = ngamsDb.ngamsDb(ngamsCfgObj.getDbServer(),
                               ngamsCfgObj.getDbName(),
                               ngamsCfgObj.getDbUser(),
                               ngamsCfgObj.getDbPassword())
    dbConObj.query("use " + ngamsCfgObj.getDbName())
    diskDic = ngamsLinuxOnlinePlugIn(dbConObj, ngamsCfgObj)
    print "Disk Dictionary = ", str(diskDic)

# --- oOo ---
```

11 *EXPERT:* The System Offline Plug-In

The purpose of the System Offline Plug-In, is to prepare the system for the Offline State, where it should be put to its 'standby state'. During this procedure, the disks could be unmounted and other actions performed like e.g. unloading of SW modules used for accessing the storage disks.

11.1 *EXPERT:* Interface of a System Offline Plug-In

The function interface of a System Offline Plug-In is the same as for the System Online Plug-In (see 10.1). A System Offline Plug-In does not return any information to NG/AMS. An exception must be thrown in case errors occur during the process of bringing the system to Offline State.

11.2 *EXPERT:* Overall Structure & Algorithm of a System Offline Plug-In

The overall structure of a System Offline Plug-In Python source is very similar to that of the DHPI. See 13.2 for further information about this issue.

11.3 *EXPERT:* Example System Offline Plug-In

In the following an example System Offline Plug-In, which is used for the moment for the NGAS installation for WFI at the La Silla 2.2m telescope. It is perhaps not a very good example of such a plug-in since most of the code is distributed in other modules. Please check the modules "ngamsEscalada6800Utils.py" and "ngamsLinuxSystemPlugInApi.py" for further information.

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsLinuxOfflinePlugIn.py,v 1.8 2002/02/19 16:54:22 safcvs Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  10/05/2001  Created.
#

"""
Module that contains System Offline Plug-In.
"""

from   ngams import *
import ngamsPlugInApi
import ngamsLinuxSystemPlugInApi, ngamsEscalada6800Utils

def ngamsLinuxOfflinePlugIn(srvObj):
    """
    Function unmounts all NGAMS disks and removes the kernel module for
    the IDE controller card.

    srvObj:        Reference to instance of the NG/AMS Server class
                   (ngamsServer).

    Returns:       Void.
    """
    rootMtPr = srvObj.getCfg().getMountRootDirectory()
    parDicOnline = ngamsPlugInApi.\
                   parseRawPlugInPars(srvObj.getCfg().getOnlinePlugInPars())
    diskDic = ngamsEscalada6800Utils.parseHtmlInfo(parDicOnline["uri"],
                                                   rootMtPr)
    parDicOffline = ngamsPlugInApi.\
                    parseRawPlugInPars(srvObj.getCfg().getOfflinePlugInPars())

    # This is only unmounting the NGAMS disks and may lead to problems
    # if someone mounts other disks off-line.
    if (parDicOffline.has_key("unmount")):
        unmount = int(parDicOffline["unmount"])
    else:
        unmount = 1
    if (unmount):
        ngamsLinuxSystemPlugInApi.ngamsUmount(diskDic)
        stat = ngamsLinuxSystemPlugInApi.rmMod(parDicOnline["module"])
        if (stat):
            errMsg = "Problem executing ngamsLinuxOfflinePlugIn! " +\
                     "The system is in not in a safe state!"
            errMsg = genLog("NGAMS_ER_OFFLINE_PLUGIN", [errMsg])
            error(errMsg)
            raise exceptions.Exception, errMsg
```

```
        msg = "Kernel module " + parDicOnline["module"] + " unloaded"
        info(1,msg)

if __name__ == '__main__':
    """
    Main function.
    """
    import sys
    import ngamsConfig, ngamsDb

    setLogCond(0, "", 0, "", 1)

    if (len(sys.argv) != 2):
        print "\nCorrect usage is:\n"
        print "% python ngamsLinuxOfflinePlugIn <NGAMS cfg>\n"
        sys.exit(0)

    ngamsCfgObj = ngamsConfig.ngamsConfig()
    ngamsCfgObj.load(sys.argv[1])
    dbConObj = ngamsDb.ngamsDb(ngamsCfgObj.getDbServer(),
                               ngamsCfgObj.getDbName(),
                               ngamsCfgObj.getDbUser(),
                               ngamsCfgObj.getDbPassword())
    dbConObj.query("use " + ngamsCfgObj.getDbName())
    ngamsLinuxOfflinePlugIn(dbConObj, ngamsCfgObj)

# --- oOo ---
```

12 *EXPERT:* The Label Printer Plug-In

The purpose of the Label Printer Plug-In is to print a label on request from NG/AMS on the label printer installed on the NGAS host. The plug-in must generate the appropriate control sequence of characters in order to request the printer to produce the label. Also other actions needed to control the printer should be taken care of by the plug-in. I.e., the plug-in could be seen as a high-level/intelligent printer driver.

12.1 *EXPERT:* Interface of a Label Printer Plug-In

A Label Printer Plug-In must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Label Printer Plug-In has an interface as shown in figure 14.1.

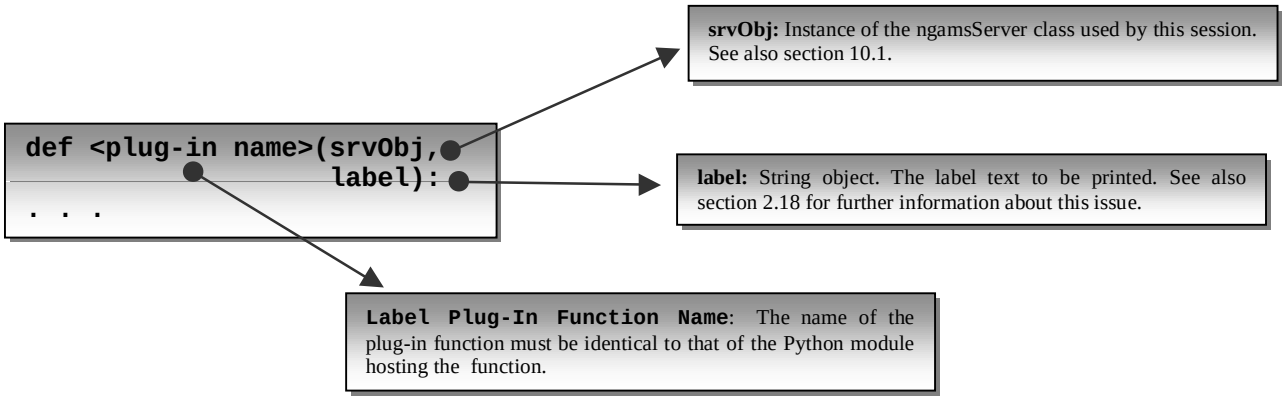


Figure 14.1: Function interface of a Label Printer Plug-In.

A Label Printer Plug-In does not return any data to NG/AMS. An exception must be thrown in case errors occur during the printing process.

12.2 *EXPERT:* Example of a Label Printer Plug-In

In the following the source code of an example is shown. This is used to control a Brother label printer (Brother P-Touch, 9200 DX).

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsBrotherPT9200DxPlugIn.py,v 1.14 2002/02/19 16:54:21 safcvs Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# awicenec/
# jknudstr  10/05/2001  Created
#

"""
This module contains a plug-in driver for printing labels on
the Brother PT-9200DX label printer.
"""

import sys, time
from   ngams import *
import ngamsPlugInApi, ngamsConfig


def genFontsDictionary(fnm):

    """
    Function reads the contents of a bitmap character file <fnm>.
    The character contents of this file has to be compliant with the  keys:

    keys = ['Header','-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
            ':', 'A','B','C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
            'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
            'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
```

```
            'z', 'Trailer']

    These keys are used to fill a dictionary with the bitmaps and can
    then be used to print strings on the Brother pTouch 9200DX
    printer.

    Synopsis:  charDict = ngamsGetCharDict(<fnm>)

    fnm:        Filename of font definition file (string).

    Returns:    Return value is a dictionary with the keys
                given above (dictionary).
    """
    keys = ['Header','-', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
            ':', 'A','B','C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
            'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
            'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
            'z', 'Trailer']

    try:
        f = open(fnm)
        charArr = f.read()
        f.close()
    except exceptions.Exception, e:
        error(str(e))
        errMsg = "Problems opening CharDict file (" + str(e) + ") "
        raise exceptions.Exception, errMsg

    charArr = charArr.split('ZG')
    charDict = {}
    i = 0
    if len(charArr) != len(keys):
        errMsg = 'Wrong number of characters in CharDict file: ' + fnm
        error(str(e))
        raise exceptions.Exception, errMsg

    for k in keys:
        if k == 'Header' or k == 'Trailer':
            charDict.update({k:charArr[i]})
        else:
            charDict.update({k:'G'+charArr[i]})   # put the G back
        charDict.update({' ':'ZZZZZZZZZZZZZ'})    # add a blank
        i = i + 1

    return charDict


def ngamsBrotherPT9200DxPlugIn(srvObj,
                               label):
    """
    Driver for printing labels on the label printer Brother PT-9200DX.

    srvObj:           Reference to instance of the NG/AMS Server
                      class (ngamsServer).

    label:            Label text to print (string).

    Returns:          Void.
    """
    plugInPars = srvObj.getCfg().getLabelPrinterPlugInPars()
    info(2,"Executing plug-in ngamsBrotherPT9200DxPlugIn with parameters: "+
         plugInPars + " - Label: " + label + " ...")
    parDic = ngamsPlugInApi.parseRawPlugInPars(plugInPars)

    # Get the font bit pattern dictionary.
    fontDic = genFontsDictionary(parDic["font_file"])

    # Generate the printer control code.
    printerCode = fontDic["Header"]
    for i in range(len(label)):
        if (not fontDic.has_key(label[i])):
            errMsg = "No font defintion for character: \"" + label[i] +\
                     "\" - in font definition file: " + parDic["font_file"] +\
                     " - cannot generate disk label: " + label
            error(errMsg)
            ngamsPlugInApi.notify(srvObj.getCfg(), NGAMS_NOTIF_ERROR,
                                  "ngamsBrotherPT9200DxPlugIn: " +\
                                  "ILLEGAL CHARACTER REQ. FOR PRINTING",
                                  errMsg)
            raise exceptions.Exception, errMsg

        printerCode = printerCode + fontDic[label[i]]
    printerCode = printerCode + fontDic["Trailer"]

    # Generate printer file, write printer control code.
    printerFilename = "/tmp/ngamsLabel_" + srvObj.getCfg().genNgasId() + ".prn"
```

```
    fo = open(printerFilename, "w")
    fo.write(printerCode)
    fo.close()

    # Write the printer code file to the device.
    res = ngamsPlugInApi.execCmd("cat "+printerFilename + " > "+parDic["dev"])
    os.system("rm -f " + printerFilename)
    if (res[0] != 0):
        errMsg = "Problem occurred printing label!"
        error(errMsg)
        ngamsPlugInApi.notify(srvObj.getCfg(), NGAMS_NOTIF_ERROR,
                              "ngamsBrotherPT9200DxPlugIn: " +\
                              "PROBLEM PRINTING LABEL", errMsg)
        raise exceptions.Exception, errMsg

    info(2,"Executed plug-in ngamsBrotherPT9200DxPlugIn with parameters: "+
         plugInPars + " - Label: " + label + " ...")


if __name__ == '__main__':
    """
    Main function.
    """
    setLogCond(0, "", 0, "", 5)
    if (len(sys.argv) != 3):
        print "\nCorrect usage is:\n"
        print "% (python) ngamsBrotherPT9200DxPlugIn <NGAMS CFG> <text>\n"
        sys.exit(1)
    cfg = ngamsConfig.ngamsConfig()
    cfg.load(sys.argv[1])
    ngamsBrotherPT9200DxPlugIn(cfg, sys.argv[2])

#
# ___oOo___
```

13 *EXPERT:* The Data Handling Plug-In - DHPI

The purpose of the Data Handling Plug-In, is to handle the archiving of data files. There are often specific aspects to take into account while archiving various kinds of data. The DHPIs make it possible to adapt NG/AMS for handling new (user specific) types of data. I.e., nothing is hard coded in the SW in connection with the data handling.

When the NG/AMS Server receives an Archive Request, a thread is spawned to handling the request. It first classifies the data and finds the appropriate Storage Set of disks on which to store the file. Subsequently it receives the data into an intermediate file with a unique name in the Staging Area on the Main Disk of the target Storage Set. The target Storage Set is determined from the NG/AMS Configuration. From the mime-type of the data a suitable Stream is found, and afterwards a suitable Storage Set.

After having received the file, the DHPI configured for handling that type of data is invoked and have to carry out data type specific tasks to be done during the archiving.

The main tasks of a DHPI are as follows:

- **Data Consistency Checking:** Usually it is advisable to carry out a check of the data before archiving it. Such a check could e.g. be to calculate the checksum of the file, or to check that certain parameters are properly set in the data file. If inconsistencies are found, the file should be moved to the Bad Files Directory on the target disk. This however, is done by NG/AMS. If a file is found to be bad, an exception should be thrown, which contains the error mnemonic "NGAMS_ER_BAD_FILE"; see the example DHPI, section 13.3 for clarification on this topic (Function: "checkCheckSum()").
- **Data Processing:** Before archiving a file, it is often necessary/required to do some processing. It could be something as simple as compressing the file, but in principle there are no limits to the kind of data processing that can be carried out. If the processing changes the mime-type of the file, it is important that the DHPI returns the proper type to NG/AMS.
- **Generating Final (Target) Filename:** The target filename of a data file may be generated from parameters in the header. The filename is composed by the Mount Point of the disk plus the Path Prefix from the configuration. How the rest of the filename is generated is up to the plug-in implementation.
- **Generating Standard DHPI Return Value:** A number of parameters like File ID, File Version, file size, Disk ID and more for the file archived must be returned to NG/AMS in order to update the NGAS DB accordingly. A convenience function provided in "ngamsPlugInApi" should be used for this.

After the DHPI has finished execution, NG/AMS will move the processed file to its final destination (which is decided by the DHPI). Also the NGAS DB is updated by NG/AMS with the information about the new file. If Replication is requested, the file is replicated and the DB updated, also with the information for the Replication File.

The DHPI is only concerned with the Main File. If a Replication File should be produced this is entirely handled by NG/AMS.

The diagram in figure 15.1 shows the main actions carried out by NG/AMS and the DHPI while handling an Archive Request. Only the main actions are shown in the figure. Behind the scenes a number of other tasks must be performed in order to archive a file properly.

As seen in figure 15.1, the handling of an Archive Request is initiated by a data provider sending an Archive Pull or an Archive Push Request to the NG/AMS Server (1). NG/AMS determines the type of data (mime-type) and receives it into the Staging Area on the Target Main Disk (2). Subsequently it invokes the DHPI (3), which does the necessary data consistency checking, processing and extraction of information from the file (4). The DHPI returns control to NG/AMS and delivers back a set of information needed by NG/AMS for the further processing of the file (5). NG/AMS stores the Main File in its final location on the Main Disk (6). Then the information about the new Main File is updated in the NGAS DB (7). If replication is enabled and a Replication Disk is defined, NG/AMS creates the Replication File (8). Afterwards the information for the Replication File is updated in the DB (9). NG/AMS can either return an immediate reply to the client issuing the Archive Request or it can return a reply when the file has been successfully (or unsuccessfully) handled.

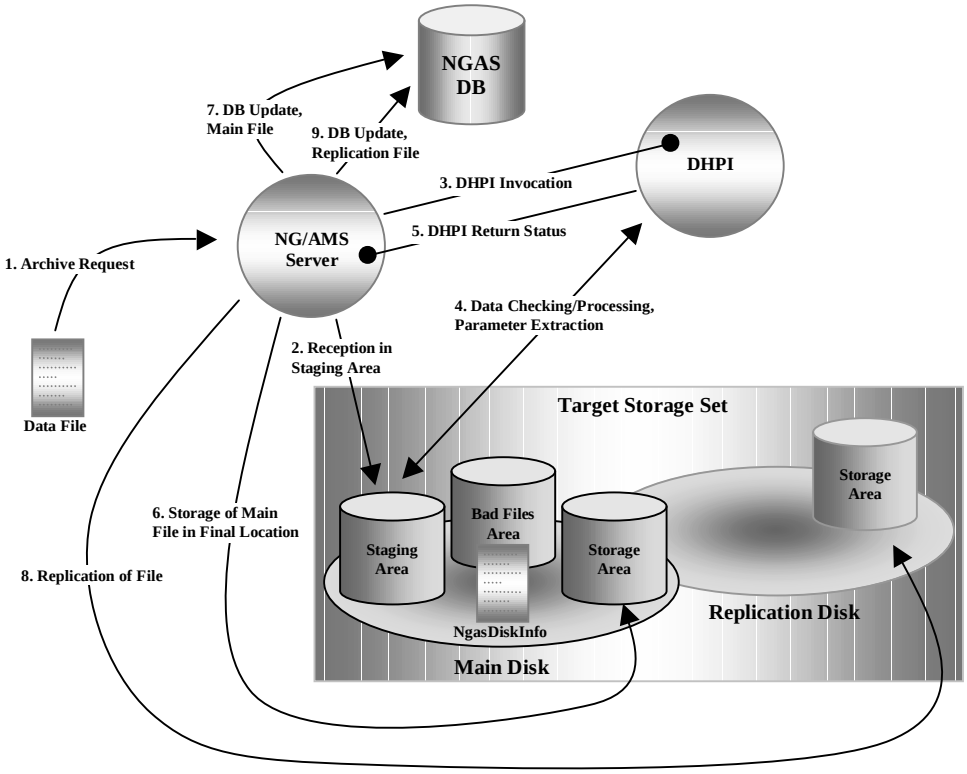


Figure 15.1: Interaction while handling an Archive Request.

Note, that the DHPI is a function running within the same Python interpreter as the NG/AMS Server process.

13.1 *EXPERT:* Interface of a DHPI

The DHPI must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual DHPI, which is invoked by NG/AMS.

A DHPI has an interface as shown in figure 15.2.

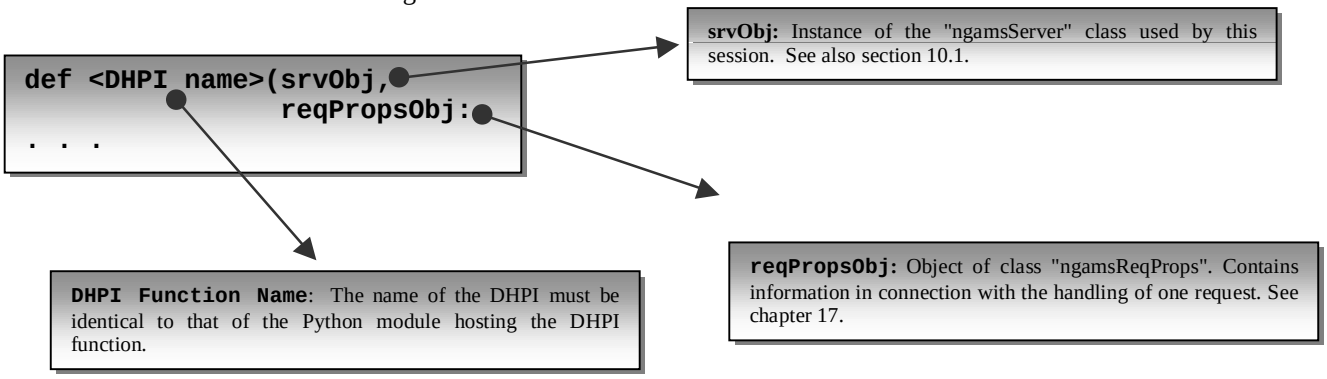


Figure 15.2: Function interface of a DHPI.

A DHPI must perform the following return when finishing execution:

```
return ngamsPlugInApi.genDhpiSuccessStat(diskId,
                                         relFilename,
                                         fileId,
                                         fileVersion,
                                         format,
                                         fileSize,
                                         uncomprSize,
                                         compression,
                                         relPath,
                                         slotId,
                                         fileExists,
                                         completeFilename)
```

The return parameters of a DHPI are as follows:

Parameter	*Type*	*Description*
diskId	String	Disk ID of file.
relFilename	String	Filename relative to mount point.
fileId	String	File ID allocated to the file by the DHPI.
fileVersion	Integer	Version of the file.
format	String.	Format (or mime-type) of the file. Only mime-types defined in the NG/AMS Configuration are accepted.
fileSize	Integer	Size of the file as it is archived.
uncomprSize	Integer	Uncompressed size of the file. I.e., if the file was compressed, this is the original size before archiving/compression.
compression	String	Compression method used to compress file. Should be the command invoked to compress the file, e.g. "compress".
relPath	String	Path relative to the mount point of the target disk.
slotId	String	Slot ID of slot in which the Main Disk is installed.
fileExists	Integer	Indicates if the file already existed on the target disk. In case yes, this should be 1, otherwise 0.
completeFilename	String	The complete name of the file as it should be. The complete name must be generated by the DHPI.

13.2 *EXPERT:* Overall Structure & Algorithm of a DHPI

The overall structure of a DHPI Python source and in particular the DHPI function itself is shown in figure 15.3.

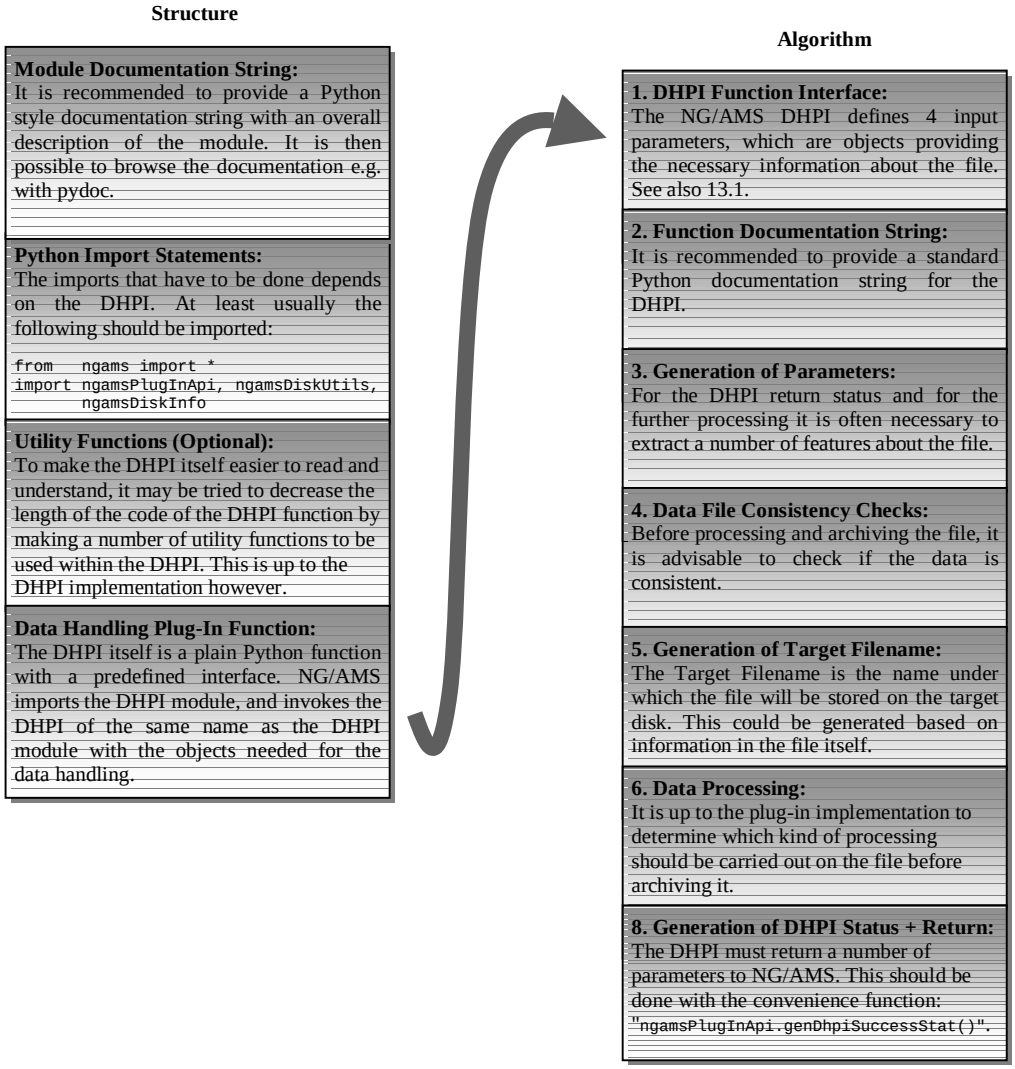


Figure 15.3: Typical structure of a DHPI module and a DHPI function.

The exact sequence of the actions performed and the actions themselves, may vary from DHPI to DHPI. I.e., maybe the data processing is done before the generation of the final target filename. In section 13.3 an example DHPI module is shown.

13.3 *EXPERT:* Example DHPI - WFI/FITS File DHPI

In the following an example DHPI, which is used for archiving FITS files at the 2.2m telescope at La Silla is shown:

```
#*******************************************************************************
# ESO/DMD
#
# "@(#) $Id: ngamsFitsPlugIn.py,v 1.42 2002/02/26 17:25:41 safcvs Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  10/05/2001  Created
#

"""
This Data Handling Plug-In is used to handle reception and processing
of FITS files.

Note, that the plug-in is implemented for the usage at ESO. If used
in other contexts, a dedicated plug-in matching the individual
context should be implemented and NG/AMS configured to use it.
"""

import os, exceptions, string
import PccUtTime, PccUtString
from   ngams import *
import ngamsPlugInApi, ngamsDiskUtils, ngamsDiskInfo


def getDpIdInfo(filename):
    """
    Generate the File ID (here DP ID) for the file.

    filename:   Name of FITS file (string).

    Returns:    Tuple containing the value of ARCFILE, the DP ID
                of the file, and the JD date. The two latter deducted from
                the ARCFILE keyword (tuple).
    """
    keyDic  = ngamsPlugInApi.getFitsKeys(filename, ["ARCFILE"])
    arcFile = keyDic["ARCFILE"][0]
    els     = string.split(arcFile, ".")
    dpId    = els[0] + "." + els[1] + "." + els[2]
    date    = string.split(els[1], "T")[0]
    # Make sure that the files are stored according to JD
    # (one night is 12am -> 12am).
    isoTime = els[1]
    ts1 = PccUtTime.TimeStamp(isoTime)
    ts2 = PccUtTime.TimeStamp(ts1.getMjd() - 0.5)
    dateDirName = string.split(ts2.getTimeStamp(), "T")[0]
    return [arcFile, dpId, dateDirName]


def checkFitsFileSize(filename):
    """
    Check if the size of the FITS file is a multiple of 2880. If this
    is not the case, we through an exception.

    filename:   FITS file to check (string).

    Returns:    Void.
    """
    if (string.split(filename, ".")[-1] == "fits"):
        size = ngamsPlugInApi.getFileSize(filename)
        testVal = (size / 2880.0)
        if (testVal != int(testVal)):
            errMsg = "The size of the FITS file issued: " + filename +\
                     ", is not a multiple of 2880! Rejecting file!"
            errMsg = genLog("NGAMS_ER_DHPI", [errMsg])
            raise exceptions.Exception, errMsg


def checkChecksum(parDic,
                  filename):
    """
    Check that the checksum of the file is correct.

    parDic:    Dictionary with disk information (ngamsPhysDiskInfo objects)
               (dictionary).

    filename:  Name of FITS file (string).

    Returns:   Void.
    """
```

```
    # Only do check if the checksum_util parameter is set.
    if (not parDic.has_key("checksum_util")): return

    # Execute the checksum routine and evaluate result.
    info(2,"Invoking checksum test utility: " + parDic["checksum_util"] +\
         " on file: " + filename)
    res = ngamsPlugInApi.execCmd(parDic["checksum_util"] + " " + filename)
    if (int(res[0]) != 0):
        errMsg = "Problem occurred invoking checksum check utility: " +\
                 parDic["checksum_util"]
        errMsg = genLog("NGAMS_ER_DHPI", [errMsg])
        error(errMsg)
        raise exceptions.Exception, errMsg
    if (res[1] != parDic["checksum_result"]):
        errMsg = "Executing checksum utility: " + parDic["checksum_util"] +\
                 " gave unexpected result. Result: [" + res[1] + "]. " +\
                 "Expected Result: [" + parDic["checksum_result"] + "]."
        errMsg = genLog("NGAMS_ER_BAD_FILE", [filename, "ngamsFitsPlugIn",
                                              errMsg])
        error(errMsg)
        raise exceptions.Exception, errMsg


# Main DHPI function.
def ngamsFitsPlugIn(srvObj,
                    reqPropsObj):
    """
    Data Handling Plug-In to handle archiving of FITS files.

    srvObj:       Reference to NG/AMS Server Object (ngamsServer).

    reqPropsObj:  NG/AMS request properties object (ngamsReqProps).

    Returns:      Standard NG/AMS Data Handling Plug-In Status
                  as generated by: ngamsPlugInApi.genDhpiSuccessStat()
                  (ngamsDhpiStatus).
    """
    stagingFilename = ""
    trgFilename = ""
    mountPoint = ""
    info(1,"Plug-In handling data for file with URI: " +
         os.path.basename(reqPropsObj.getFileUri()))
    diskInfo = reqPropsObj.getTargDiskInfo()
    parDic = ngamsPlugInApi.parseDhpiPlugInPars(srvObj.getCfg(),
                                                reqPropsObj.getMimeType())
    # If the file is already compressed, we have to decompress it.
    tmpFn = reqPropsObj.getStagingFilename()
    if ((tmpFn.find(".Z") != -1) or (tmpFn.find(".gz") != -1)):
        ngamsPlugInApi.execCmd("gunzip " + tmpFn)
        reqPropsObj.setStagingFilename(os.path.splitext(tmpFn)[0])
    stagingFilename = reqPropsObj.getStagingFilename()
    comprExt = ""
    if (parDic.has_key("compression")):
        if (string.split(parDic["compression"], " ")[0] == "compress"):
            comprExt = "Z"
        elif (string.split(parDic["compression"], " ")[0] == "gzip"):
            comprExt = "gz"

    # Check file (size + checksum).
    checkFitsFileSize(stagingFilename)
    checkChecksum(parDic, stagingFilename)

    # Get various information about the file being handled.
    dpIdInfo = getDpIdInfo(stagingFilename)
    dpId = dpIdInfo[1]
    dateDirName = dpIdInfo[2]
    fileVersion, relPath, relFilename,\
                 complFilename, fileExists =\
                 ngamsPlugInApi.genFileInfo(srvObj.getDb(), srvObj.getCfg(),
                                            diskInfo, stagingFilename, dpId,
                                            dpId, [dateDirName], [comprExt])

    # If a compression application is specified, apply this.
    uncomprSize = ngamsPlugInApi.getFileSize(stagingFilename)
    if (parDic["compression"] != ""):
        info(2,"Compressing file using: " + parDic["compression"] + " ...")
        exitCode, stdOut, stdErr =\
                  ngamsPlugInApi.execCmd(parDic["compression"] +\
                                         " " + stagingFilename)
        if (exitCode != 0):
            errMsg = "ngamsFitsPlugIn: Problems during data handling! " +\
                     "Compressing the file failed"
            raise exceptions.Exception, errMsg
        stagingFilename = stagingFilename + "." + comprExt
        # Remember to update the Temporary Filename in the Request
        # Properties Object.
```

```
        reqPropsObj.setStagingFilename(stagingFilename)
        info(2,"File compressed")

    # Generate status.
    info(4,"Generating status ...")
    format = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), stagingFilename)
    fileSize = ngamsPlugInApi.getFileSize(stagingFilename)
    info(3,"DHPI finished processing of file")
    return ngamsPlugInApi.genDhpiSuccessStat(diskInfo.getDiskId(), relFilename,
                                             dpId, fileVersion, format,
                                             fileSize, uncomprSize,
                                             parDic["compression"], relPath,
                                             diskInfo.getSlotId(), fileExists,
                                             complFilename)

#
# ___oOo___
```

14 *EXPERT:* The Data Processing Plug-In - DPPI

The purpose of the Data Processing Plug-In (DPPI) is to provide a specific type of processing to be applied on a specific type of data when data is being retrieved from an NGAS system. Processing could be as trivial as simply uncompressing a data file, which is stored in compressed format. It could also be far more complex and involve advanced image processing and parameter extraction. How the DPPI actually processes the data, is left up to the DPPI implementation. The DPPI only has to obey the set of rules for interfacing as for any other plug-in defined for NG/AMS.

14.1 *EXPERT:* Interface of a DPPI

The DPPI must be contained in a Python module (file), which has a function of the same name as the module. The latter is the actual DPPI, which is invoked by NG/AMS.

A DPPI has an interface as shown in figure 16.1.

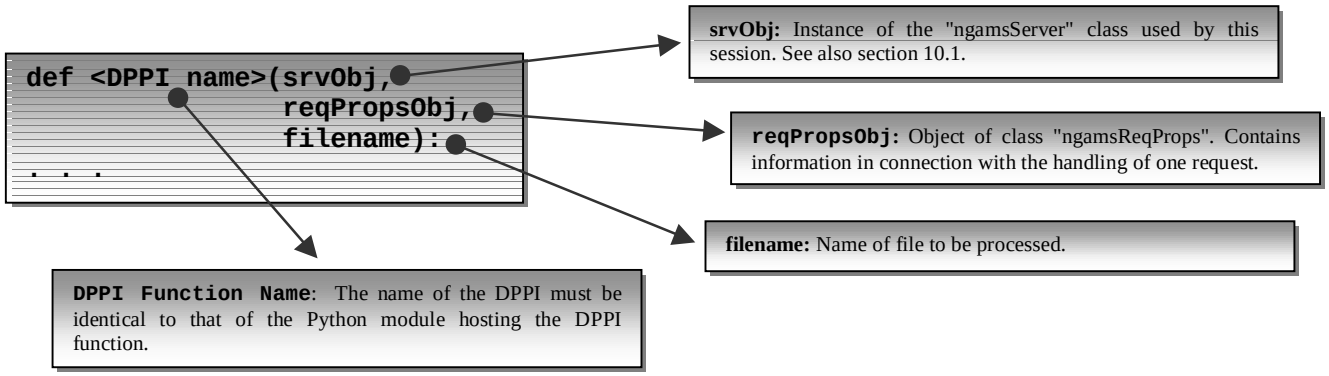


Figure 16.1: Function interface of a DPPI.

A DPPI must return an object of the type "ngamsDppiStatus". This again contains one or more objects of the type "ngamsDppiResult", which each refer to result data or contains the result of the processing. This means that it is possible to produce several results in a DPPI, and to have these send back to the requestor[4]. The concept of the DPPI return object is shown in figure 16.2.

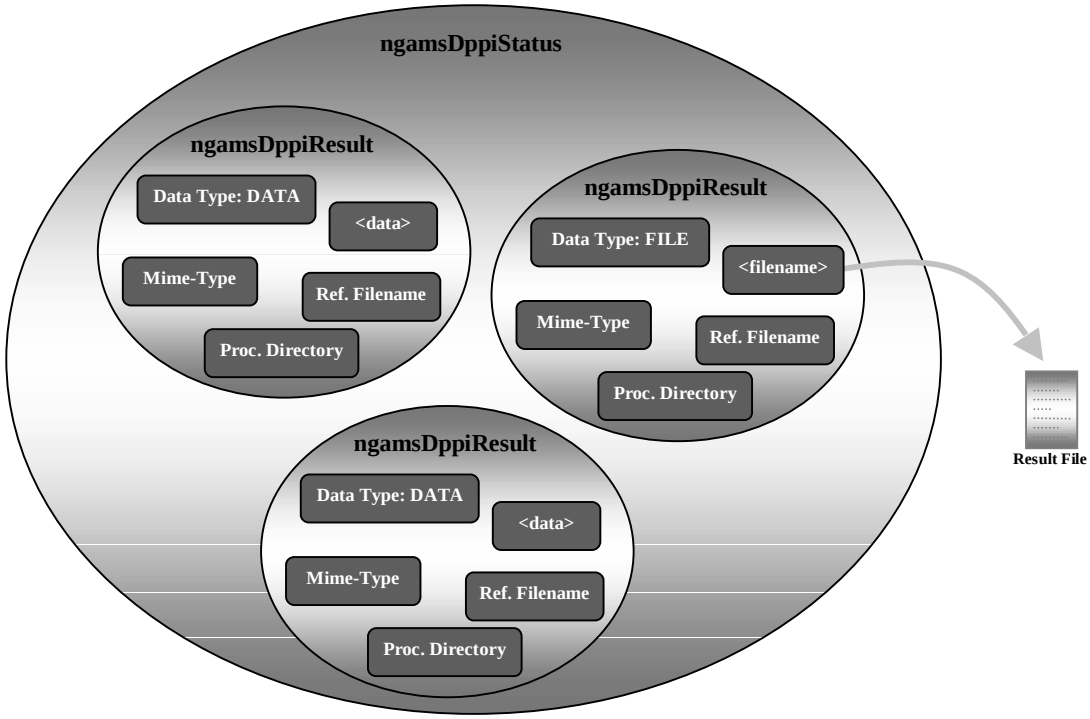


Figure 16.2: DPPI return object structure.

As shown in figure 16.2, the "ngamsDppiStatus" object can contain an arbitrary number of "ngamsDppiResult" objects, each containing the information of one sub-result. As can be seen in the figure, the data of a sub result can either be contained directly in the ngamsDppiResult object, or the data can be stored in an external file, which is referred to by the object. Whether or not to use the one or the other depends on the nature of the data. If the result consists of a smaller amount of non-binary data it is more convenient to store the data internally to avoid having to create, access and delete the result files. For larger amounts of result data and for binary data, it is recomended to use an external result file. See chapter 17 for more information about these classes.

14.2 *EXPERT:* Example DPPI

In the following a very trivial example of a DPPI is shown. It is used to decompress files, which have been archived in compressed format.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsEsoArchDppi.py,v 1.6 2002/02/20 16:39:54 safcvs Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  08/01/2002  Created
#

from ngams import *
import ngamsPlugInApi, ngamsDppiStatus


def ngamsEsoArchDppi(srvObj,
                     reqPropsObj,
                     filename):
    """
    This DPPI performs the processing neccessary for the files
    requested from the ESO Archive (by the Data Requestor).
```

[4] For actually supporting completely latter, NG/AMS needs to be extended to be able to return replies making use of the "multipart/mixed" mime-type as known from e.g. emails. This is foreseen to be supported soon.

```
    srvObj:         Reference to instance of the NG/AMS Server
                    class (ngamsServer).

    reqPropsObj:    NG/AMS request properties object (ngamsReqProps).

    filename:       Name of file to process (string).

    Returns:        DPPI return status object (ngamsDppiStatus).
    """
    statusObj = ngamsDppiStatus.ngamsDppiStatus()

    # Decompress the file if the last extension is "Z".
    if (filename.split(".")[-1] == "Z"):
        procFilename, procDir = ngamsPlugInApi.prepProcFile(srvObj.getCfg(),
                                                            filename)
        exitCode, stdOut, stdErr = ngamsPlugInApi.\
                                   execCmd("uncompress " + procFilename)
        if (exitCode != 0):
            errMsg = "ngamsEsoArchDppi: Problems during data handling! " +\
                     "Decompressing the file: " + filename + " failed. " +\
                     "Error message: " + str(stdErr)
            raise exceptions.Exception, errMsg
        resFilename = procFilename[0:-2]
    else:
        resFilename = filename
        procDir = ""
    mimeType = ngamsPlugInApi.determineMimeType(srvObj.getCfg(), resFilename)
    resObj = ngamsDppiStatus.ngamsDppiResult(NGAMS_PROC_FILE, mimeType,
                                             resFilename, resFilename, procDir)
    statusObj.addResult(resObj)

    return statusObj

#
# ___oOo___
```

15 *EXPERT:* The Data Checksum Plug-In

The Data Checksum Plug-In is a simple plug-in used to generate the checksum value for a data file being archived. This value is writte in the record for the file in the NGAS DB, and used later on to check periodically if the file is in a 'good condition'. I.e., that it is not damaged or corrupted in any way. The Data Checksum Plug-In is invoked by NG/AMS after the DHPI has finished the data type specific processing.

15.1 *EXPERT:* Interface of a Data Checksum Plug-In

The plug-in must be contained in a Python module, which has a function of the same name as the module. The latter is the actual plug-in, which is invoked by NG/AMS. A Data Checksum Plug-In has an interface as shown in figure 15.1.

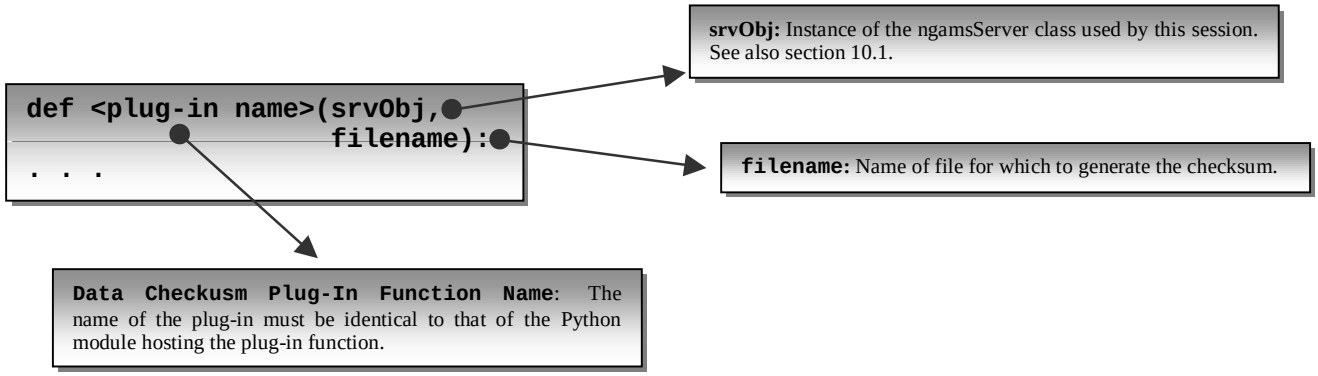


Figure 15.1: Function interface of a DCPI.

A Data Checksum Plug-In must return the calculated checksum value as a string.

15.2 *EXPERT:* Example Data Checksum Plug-In

In the following the source code of a small example Data Checksum Plug-In is shown. It generates the checksum based on routines built-into Python.

```
#*******************************************************************************
# ESO/DFS
#
# "@(#) $Id: ngamsGenCrc32.py,v 1.9 2002/02/27 17:18:25 safcvs Exp $"
#
# Who       When        What
# --------  ----------  --------------------------------------------------------
# jknudstr  23/01/2002  Created
#

import binascii
import pcc, PccUtTime
from   ngams import *

def ngamsGenCrc32(srvObj,
                  filename):
    """
    Plug-in to generate CRC-32 checksum for an archived data file.

    srvObj:       Reference to instance of NG/AMS Server class (ngamsServer).

    filename:     Name of file to generate checksum for (string).

    Returns:      CRC-32 checksum for file (string).
    """
    fo = open(filename, "r")
    buf = fo.read(65536)
    crc = 0
    while (buf != ""):
        crc = binascii.crc32(buf, crc)
        buf = fo.read(65536)
    fo.close()
    return str(crc)

#
# ___oOo___
```

16 The NG/AMS Status XML Document

The NG/AMS Status Document is used in various contexts, either as the complete status or as partial status for a specific context. For instance, in the reply of most commands, a small status is given indicating if the command was executed successfully, and in case not, indicating the error that occurred.

16.1 *EXPERT:* NG/AMS Status DTD

The NG/AMS Status Document is based on the NG/AMS base DTD described in section 4.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % XmlStd SYSTEM "XmlStd.dtd">
%XmlStd;
<!ENTITY % NgamsInternal SYSTEM "ngamsInternal.dtd">
%NgamsInternal;

<!--
  E.S.O.

  Who        When        What
  ********   **********  ************************************************
  jknudstr   04.04.2001  Created
  ************************************************************************
  ngamsStatus.dtd defines the contents and lay-out of the
  NG/AMS Status Report.

  Consult the DTD ngamsInternal.dtd which contains the actual definition
  of the elements of the NGAMS Status.
  -->


<!--
  The NgamsStatus element is the root element of the NGAMS Reply Document.
  -->
<!ELEMENT NgamsStatus (Header, Status, NgamsCfg?, DiskStatus*)>


<!--
  The Status Element is used to generate a status with log
  information. It can contain an arbitrary number of log elements
  (defined in the LogMlBase DTD).

  Attributes:
    Date:     Date this Status Element was generated.

    Version:  Version of NG/AMS generating the status.

    HostId:   Name of host where the NG/AMS Server is running.

    Status:   Overall status of the status information. Can be used to
              signal if errors are contained in the Status Elements.

    Message:  Message generated by the NG/AMS Server.

    State:    State of the NG/AMS Server.

    SubState: Sub-State of the NG/AMS Server.
  -->
<!ELEMENT Status  (Trace | Debug | Info | Warning | Error | Alarm | Archive)*>
<!ATTLIST Status  Date             CDATA             #REQUIRED
                  Version          CDATA             #REQUIRED
                  HostId           CDATA             #REQUIRED
                  Status           (OK|FAILURE|-)    "-"
                  Message          CDATA             #IMPLIED
                  State            (ONLINE|OFFLINE)  "OFFLINE"
                  SubState         (IDLE|BUSY)       "IDLE">


<!--
  The NgamsCfg Element contains the configuration used by NG/AMS.
  -->
<!ELEMENT NgamsCfg (Ngams, Server?, Db?, StorageSet*, FileHandling?,
                    Stream*, Monitor?, Log?)>


<!--
  The DiskStatus Element contains the status for each disk and the
  status for the files stored on an HDD.

  Attributes:
    DiskId:             Unique ID for the HDD.
```

```
    Archive:            ID of the archive to which this disk belongs

    InstallationDate:   Date (ISO-8601) the disk was prepared.

    Type:               Type of the HDD.

    Protected:          Indicates if the disk is marked as protected. I.e.
                        the disk is installed in a central NGAS system
                        and the contents of the DB/HDD should not be changed.

    LogicalName:        Logical (human readable/memorable) name of
                        the HDD.

    AssociatedDiskId:   ID of the associated disk (Replication Disk
                        in case of a Main Disk and vice versa).

    MainDisk:           Indicates if the HDD is the Main Disk or the
                        Replication Disk (0|1).

    HostId:             Name of the host where the HDD is installed.

    SlotId:             Slot ID (number) in which the HDD is installed.

    Mounted:            Indicates if the HDD is mounted in this NGAS
                        System (0|1).

    MountPoint:         Indicates the mount point for the HDD (path).

    NumberOfFiles:      Number of files stored on the disk.

    AvailableMb:        Indicates the capacity of the HDD (MB).

    BytesStored:        Bytes stored on the HDD.

    Completed:          Indicates if the disk is full - no more data
                        should be stored on this disk.

    Checksum:           Checksum for the data on the disk.

    TotalDiskWriteTime: Total time in seconds used for writing the
                        bytes stored on this disk (s).
  -->
<!ELEMENT DiskStatus (FileStatus*)>
<!ATTLIST DiskStatus DiskId              CDATA  #REQUIRED
                     Archive             CDATA  #REQUIRED
                     InstallationDate    CDATA  #REQUIRED
                     Type                CDATA  #REQUIRED
                     Protected           (0|1)  #REQUIRED
                     LogicalName         CDATA  #REQUIRED
                     AssociatedDiskId    CDATA  #REQUIRED
                     MainDisk            (0|1)  #REQUIRED
                     HostId              CDATA  #REQUIRED
                     SlotId              CDATA  #REQUIRED
                     Mounted             (0|1)  #REQUIRED
                     MountPoint          CDATA  #REQUIRED
                     NumberOfFiles       CDATA  #REQUIRED
                     AvailableMb         CDATA  #REQUIRED
                     BytesStored         CDATA  #REQUIRED
                     Completed           (0|1)  #REQUIRED
                     CompletionDate      CDATA  #REQUIRED
                     Checksum            CDATA  #REQUIRED
                     TotalDiskWriteTime  CDATA  #REQUIRED>


<!--
  The FileStatus Element contains the status of each file stored on
  the individual HDD.

  Attributes:
    FileName:              Name of the file (relative to the disk
                           mount point).

    FileId:                ID of the file, e.g. DP ID for ESO FITS files.

    Format:                Mime-type of file.

    FileSize:              Size of the file (bytes).

    UncompressedFileSize:  Size of file uncompressed (bytes). Same as
                           FileSize if uncompressed.

    Compression:           Type of compression applied on the file.

    IngestionDate:         Date the file was ingested (ISO-8601).
  -->
<!ELEMENT FileStatus EMPTY>
```

```
<!ATTLIST FileStatus FileName              CDATA        #REQUIRED
                     FileId                CDATA        #REQUIRED
                     Format                CDATA        #REQUIRED
                     FileSize              CDATA        #REQUIRED
                     UncompressedFileSize  CDATA        #REQUIRED
                     Compression           CDATA        #REQUIRED
                     IngestionDate         CDATA        #REQUIRED
                     Ignore                (0|1)        #REQUIRED
                     Checksum              CDATA        #REQUIRED
                     ChecksumPlugIn        CDATA        #REQUIRED
                     FileStatus            CDATA        #REQUIRED>

<!-- oOo -->
```

16.2 NGAS Disk Info Status - Example

The following is an example of an NgasDiskInfo file. Such XML status documents are stored on each NGAS disk.

```
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-02-20T16:39:37.314" HostId="acngast1" Message="Disk status file"
          Version="v1.5/2002-02-19T16:54:21"/>
  <DiskStatus Archive="ESO-ARCHIVE" AssociatedDiskId="IBM-DTLA-305040-YJ0YJ075523"
              AvailableMb="39223" BytesStored="3145220" Checksum="" Completed="0"
              CompletionDate="" DiskId="IBM-DTLA-305040-YJ0YJ070913"
              InstallationDate="2002-02-19T13:13:03.000"
              LogicalName="LS-FitsStorage4-M-000001" Manufacturer="IBM" NumberOfFiles="38"
              Protected="0" TotalDiskWriteTime="0.6526222229" Type="MAGNETIC DISK/ATA"/>
</NgamsStatus>
```

16.3 NGAS File Info Status - Example

The following is an example of a File Info Status document, which is generated e.g. when archiving a file or when issuing a "STATUS?file_id=<file ID>" request:

```
<?xml version="1.0" ?>
<NgamsStatus>
  <Status Date="2002-03-06T10:35:43.736" HostId="jewel67"
          Message="Successfully handled command STATUS" State="ONLINE" Status="SUCCESS"
          SubState="IDLE" Version="v1.5/2002-02-19T16:54:21"/>
  <DiskStatus Archive="ESO-ARCHIVE" AssociatedDiskId="IC35L080AVVA07-0-VNC400A4C1622A"
              AvailableMb="264" BytesStored="81945254726" Checksum="0" Completed="1"
              CompletionDate="" DiskId="IC35L080AVVA07-0-VNC400A4C1607A" HostId="jewel67"
              InstallationDate="2002-02-14T14:18:49.000" LogicalName="LS-FitsStorage3-M-000027"
              Manufacturer="IBM" MountPoint="/NGAS/data2" Mounted="1" NumberOfFiles="1211"
              Protected="1" SlotId="2" TotalDiskWriteTime="8556.87317019" Type="IC35L080AVVA07">
    <FileStatus Checksum="2131368935" ChecksumPlugIn="ngamsGenCrc32" Compression="compress -f"
                FileId="WFI.2002-02-22T01:33:54.081"
                FileName="saf/2002-02-21/WFI.2002-02-22T01:33:54.081.fits.Z"
                FileSize="120669633" FileStatus="00000000" FileVersion="1" Format="ngas/fits"
                Ignore="0" IngestionDate="2002-02-22T01:46:23.000"
                UncompressedFileSize="141554880"/>
  </DiskStatus>
```

17 *EXPERT:* The NG/AMS Python Modules

In this chapter an overview of the NG/AMS Python modules, classes, functions and 'constants' is given. It is not the intention to provide the complete documentation for all this. This is contained as inline Python documentation in the Python code and can be browsed online. See section 17.1 for a description of how to do this. The purpose of this chapter is merely to provide an overview of the code.

For the basic usage of NG/AMS it is normally not necessary to have a deep knowledge about the internals of the SW. However, when developing the various types of plug-ins, which must be provided to adapt NG/AMS to various specific contexts, it is an advantage, and in some cases crucial, to have some insight in and overview of the SW and the classes and features available.

17.1 *EXPERT:* Online Browsing of NG/AMS Inline Python Documentation

It is possible to browse online the Python documentation contained in the NG/AMS Python source code files. This provides an acurate and comprehensive description of all classes, methods and functions. The following notation has been used to document the interfaces of methods and functions:

```
def notify(ngamsCfgObj,
           type,
           subject,
           msg):
    """
    Send a notification e-mail to a subscriber about an event happening.

    ngamsCfgObj:   Reference to object containing NG/AMS configuration file (ngamsConfig).

    type:          Type of Notification (See NGAMS_NOTIF_* in ngams).

    subject:       Subject of message (string).

    msg:           Message to send (string).

    Returns:       Void.
    """
    <code>
```

First in the description of a method/function, a small description of the task performed by the method is provided. After that the input parameters are listed. After the description of each parameter the type of the parameter is indicated in paranthesis. The return value is also given in connection with the "Returns:" tag.

The documentation can be browsed in an easy manner by using the documentation generator provided together with the Python package ("pydoc"). This can also be used as an HTTP server, e.g.:

```
arcus1 jknudstr:~/dev/ngams 65 > pydoc -p 7878 &
[2] 15578
arcus1 jknudstr:~/dev/ngams 66 > pydoc server ready at http://localhost:7878/
```

The NG/AMS documentation can now be accessed online via the URL, e.g.:

http://arcus1.hq.eso.org:7878/ngams.html

The pydoc utility provides a convenient way of browsing the documentation, and generates the documentation online. It locates the NG/AMS module from the "PYTHON_PATH" environment variable.

17.2 *EXPERT:* NG/AMS Modules

The main NG/AMS project module contain the following files and modules (only items of interest in this context are listed:

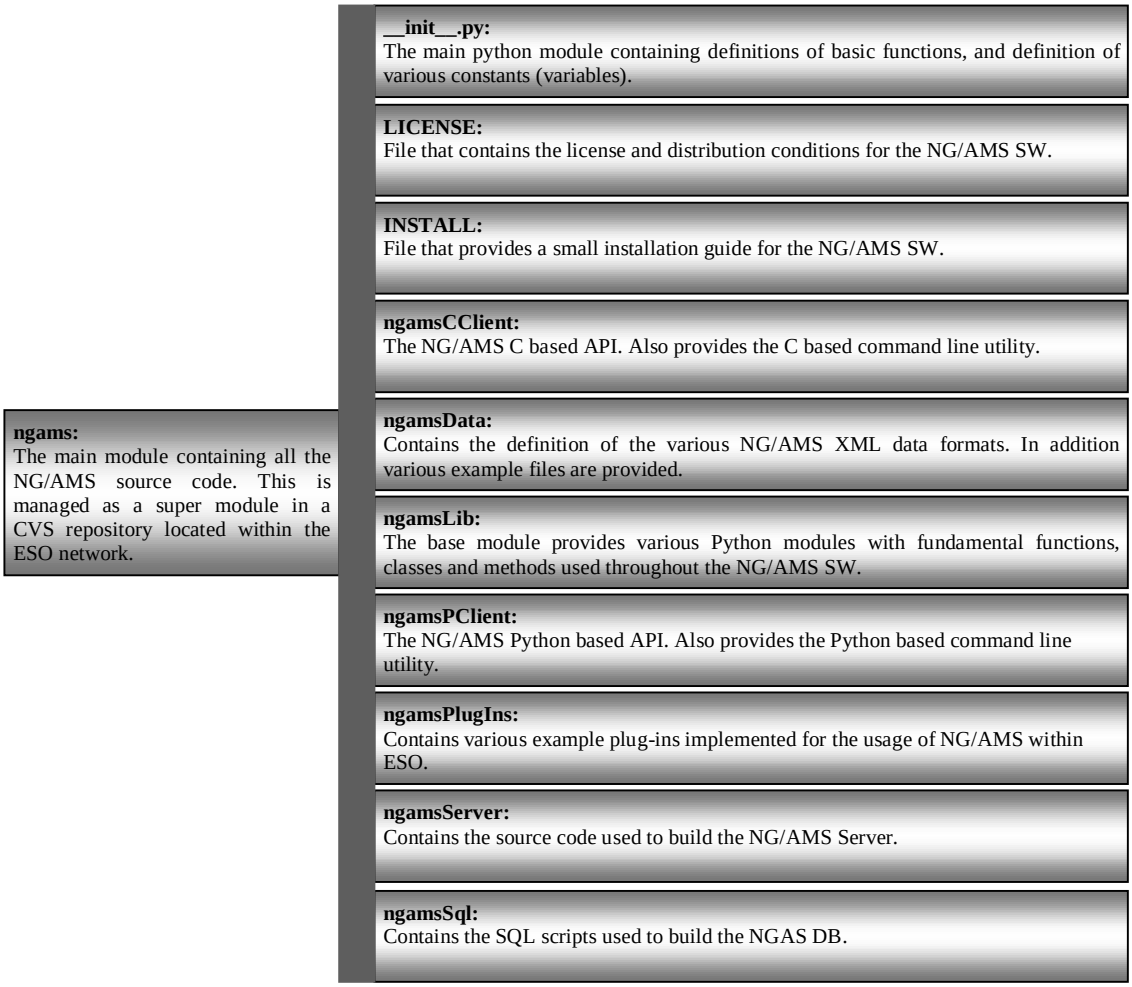


Figure 17.1: NG/AMS module structure.

The "ngamsLib" module is the one, which a plug-in developer mostly will be concerned with, although some knowledge about the NG/AMS Server class (and Python module) is also needed. In the following some components of interest for plug-in development are described.

17.2.1 *EXPERT:* Module: "ngamsLib"

The Python modules of the ngamsLib SW module provides the following Python modules:

Python Module	*Class*	*Description*
ngamsConfig.py		Contains the code for the "ngamsConfig" class together with other classes used in connection with the NG/AMS Configuration. This is all used to handle the configuration programmatically.
	ngamsStorageSet	Class used to manage the information in connection with one Storage Set from the NG/AMS Configuration.
	ngamsStream	Class used to manage the information in connection with a Stream Definition from the NG/AMS Configuration.
	ngamsConfig	Class used to handle the information in the NG/AMS Configuration. It is possible to load and save the configuration file, as well as to setting and getting all properties of the configuration. It is also possible to generate an XML document of the configuration contained in the object.
ngamsDb.py		The module provides the class "ngamsDb", which is used to access the NGAS DB. All DB access should be performed through this class. This therefore contains all the necessary SQL queries used by the NG/AMS SW.
ngamsDhpiStatus.py		The module provides the class "ngamsDhpiStatus", which is used to handle the status information from the execution of a Data Handling Plug-In (see chapter 13). An instance of this class is returned by a DHPI to the NG/AMS Server.

ngamsDiskInfo.py		Provides the class "ngamsDiskInfo", which is used to handle all the information in connection with an NGAS disk. The object can also contain information about the files on the disk. This is stored internally as "ngamsFileInfo" objects. It is possible to generate an NG/AMS XML Status document from the contents of the object.
ngamsDiskUtils.py		Functions used to carry out the handling/management of the disks installed. Among this are function to extract the information about the disk configuration, and a function to check the accessibility of the disks installed.
ngamsDppiStatus.py		The module provides the class "ngamsDppiStatus", which is used to handle the status information from the execution of a Data Processing Plug-In (see chapter 14). An instance of this class is returned by a DPPI to the NG/AMS Server.
	ngamsDppiResult	Class that contains a sub-result from a DPPI execution.
	ngamsDppiStatus	Class that contains the resulting data from a DPPI execution.
ngamsFileInfo.py		The module provides the class "ngamsFileInfo", which is used to handle all the information in connection with a file, which has been archived in an NGAS System. It is possible to generate an XML document from the contents of the object.
ngamsLib.py		Contains various basic convenience functions used throughout the NG/AMS SW.
ngamsPhysDiskInfo.py		Provides the class "ngamsPhysDiskInfo", which is used to manage the 'physical information' about a disk extracted by the System Online Plug-In (see chapter 10).
ngamsPlugInApi.py		Modules that provides various utility functions to be used for implementing plug-ins. It is recommended only to use the functions contained in this module for implementing the plug-in, apart from varios classes like ngamServer, ngamsDb and ngamsConfig.
ngamsReqProps.py		Module that provides the object "ngamsReqProps", which is used to keep a record of actions carried out during the handling of a request.
ngamsStatus.py		Provides the class "ngamsStatus", which is used to handle the information in connection with a status generated for NG/AMS.
ngamsUrlLib.py		Modules that provides a small class "ngamsURLopener", which is used to access URLs in a transparent manner.

17.2.2 *EXPERT:* Module: "ngamsServer"

The Python modules of the ngamsServer SW module provides the following Python modules:

Python Module	*Class*	*Description*
ngamsArchiveUtils.py		Contains various functions used by the NG/AMS Server class to handle the archiving of files.
ngamsBackLogBuffering.py		Contains various functions to handle the Back-Log Buffering of data files.
ngamsCmdHandling.py		Contains functions to handle the commands supported by the NG/AMS Server.
ngamsRetrieveProc.py		Contains functions to handle the retrieval and processing of data files.
ngamsServer.py		Contains the NG/AMS Server class, "ngamsServer", and some other classes and functions for the basic handling of requests.
ngamsSrvThreads.py		Contains the code for the NG/AMS Server threads (Janitor Thread, Data Consistency Check Thread) and other functions used in this connection.

18 *EXPERT:* Installation

The installation of an NG/AMS Server *can* be a relatively straightforward and simple procedure if it is not necessary to create an adapted installation for a specific context. Under normal circumstances, the most complex action in this connection might be to configure the server properly. The steps to carry out to obtain a running NG/AMS installation are as follows:

Step/Action	*Description*
Verify Sybase Installation[5]	Verify that a Sybase server is available (Sybase ASE 12). Also check that it is possible to connect to the server from the NGAS host (with isql). If this is not possible an entry should be added in the "$SYBASE/interfaces" file. It should also be verified that the libraries "libct.a/libct.sl|.so" are available in "$SYBASE/lib".
Install Python, Check Existing Python Installation	The present version of Python required for NG/AMS is 2.1 (2.1.1). Check that the proper version is installed. If the wrong version is installed, or if Python is not installed at all, it should be downloaded from **http://www.python.org** and installed according to the instructions. Check in particular that the Sybase Python module is available.
Get the NG/AMS Python SW	Get the sources of the NG/AMS SW. This can be requested by contacting: **ngast@eso.org**.
Install NG/AMS SW + Configure the Environment	Install the sources simply by copying the NG/AMS root module directory "ngams" to a path contained within the "PYTHON_PATH" list of paths, or add the new location of "ngams" in the "PYTHON_PATH" variable. The NG/AMS C-API should also be compiled and installed. This is done by entering in the directory "ngams/ngamsCClient" and typing "make clean all". The binary "ngamsCClient" should be installed in a 'bin' directory for global access. The "ngams.h" and "libngams.a" files should be copied to an area where global accessible (if needed for application development). It could also be chosen to make the NG/AMS Server source file ("ngams/ngamsServer/ngamsServer.py") executable and global accessible. The same goes for the NG/AMS Python API ("ngams/ngamsPClient/ngamsPClient.py").
Prepare Sybase DB	Prepare the NGAS DB in the Sybase DB server. A user to be used by the NG/AMS when connecting to the DB should be created, e.g.: "ngas". The NGAS tables must also be created. This should be done using the SQL scripts contained in "ngams/ngamsSql". The scripts are called: "ngamsCreateTables.sql" and "ngamsCreateHostsTable.sql". The latter should only be executed if it is desirable to use the "ngas_hosts" table (see section 6.1). The scripts can be executed using "isql".
Prepare NG/AMS Configuration	Use possibly as a template configuration the configuration example file provided within the NG/AMS SW package ("ngams/ngamsData/ngamsCfgNau.xml"). Go carefully through the list of parameters and configure these according to the description provided in chapter 4).
Prepare Plug-Ins	Prepare the necessary plug-ins needed for operating NG/AMS. The plug-ins to consider first are the System Online and Offline Plug-Ins; see the chapters 10 and 11. In addition the DHPI for each type of data to be handled (archived); see chapter 13. If it is desirable to calculate a checksum for the data files being archived, a Data Checksum Plug-In must be provided; see chapter 15. If data should be processed, a DPPI should be provided for each type of processing offered by the system; see chapter 14. If labels for the disk cases should be generated, a Label Printer Plug-In must be provided as well; see chapter 12. Example implementations of all of these types of plug-in are provided within the NG/AMS package ("ngams/ngamsPlugIns"). Note that all plug-ins provided should be made available in a path pointed to by the "PYTHON_PATH" variable.
Launch Server in Simulation Mode	The first time when the NG/AMS Server is started after doing all the necessary configuring, it may be convenient to start it manually in an xterm in the Verbose Mode (Verbose Level 3 or 4); see also section 3.1. This could be done in Simulation Mode to first get the basic things straightened out. If the server encounters problems, it will bail out and report these on stdout. The switching on/off of the Simulation Mode/Normal Mode must be done in the NG/AMS

[5] In the present release only Sybase is supported. On request it may be that other DBMS' will be supported in the future.

	Configuration. It could be tried to issue some commands like ARCHIVE and RETRIEVE to verify the proper functioning.
Launch server in real mode	When the server is running properly in Simulation Mode, it could be tried to switch to the Normal Mode (in the configuration), and try the same as described in the previous step.
Decide how to Start the Server	When the server can be executed and is operating correctly, it should be decided how it should be started. Under normal circumstances it should be started when the host on which it is running is booting, and run as a daemon. I.e., the start-up scripts on the host should be configured accordingly.
Handling of Local Log Files	If a Local Log File is generated, it should be considered that this will continuously grow in size. The speed with which it will be growing, depends on the Log Level selected. If it is desirable to keep the log files, a DHPI to handle this could be provided for NG/AMS and a cron job launched periodically to archive the log file into NG/AMS and subsequently to delete it. If it is not desirable to preserve the information, the file could be deleted periodically. This however, is up to the people responsible for the individual installation how to handle this.
Configuring of Security Mechanisms	Since no security mechanims are provided at the level of NG/AMS to prevent 'intruders' to connect to the server, such mechanisms should be put in place at the level of the operating system or network. It is up to the responsibles for the security in connection with IT services to decide how to implement this.
Setting Up of Multisite DB Environment	If an organization is running a distributed NGAS system, whereby data e.g. are produced on several remote sites, and are made available in a central archive, considerations should be done as how to set up the DB infrastructure. It might be most logical to have the central/reference DB in connection with the central archive, and to set up replication from the various remote sites to the central archive DB.

As can bee seen above, the installation in the worst-case may be a quite complex procedure. It is therefore not feasible to provide more information in the NG/AMS User's Manual about this. In case of problems or questions it is suggested to contact: **ngast@eso.org** for advice and help on how to approach this matter.

19 NG/AMS Commands

This chapter contains a detailed reference to the commands supported by the NG/AMS Server. All the commands are listed and explained, and the command parameters in connection with these are described.

In the following the parameters in connection with each command are listed individually. The actually URL that is issued to NG/AMS has an appearance as follows:

http://<host>:<port>/<command>[?<parameter>=<value>][&<parameter>=<value>]

Also the possible combinations of parameters or restrictions in their usage is described.

Using the NG/AMS APIs (chapters 7 and 8) or the NG/AMS Python or C based command line utilities (serctions 3.2 and 3.3), the user is assisted in applying the proper parameters. It is recommended to use these when communicating with the NG/AMS Server.

19.1 ARCHIVE Command - Archiving Data Files

The ARCHIVE command is used to archive data files. The ARCHIVE command accepts the following parameters:

Parameter	*Mandatory*	*Description*
-file_uri <URI>	Yes	The parameter is used to specify the location of the file. In case of an Archive Push Request, NG/AMS uses the given URI, to determine the mime-type of the file. In addition, the temporary filename in the Staging Area is based on the filename (without the path) given in the URI. For an Archive Push Request, the URI is the location (URL) where the file can be picked up by the NG/AMS Server. The location must then be accessible from the NG/AMS Server either via HTTP (http://…), FTP (ftp://…) or directly as file (file://…).
-wait 0|1	No	With this parameter it is possible to specify if the NG/AMS Server should send back an immediate reply (wait=0) when handling an Archive Request, or if a reply should be sent after the request has been handled (wait=1). In the former case, the client will not know if the Arhive Request was handled successfully. The default behavior of the server is to send the *reply* after the Archive Request has been handled.
-mime_type <mime-type>	No	If the File URI of an Archive Request does not reveal the mime-type of the file to be archived, the mime-type should be specified in the archive request. This makes the handling of the request much more efficient. An example of such a case is given in Example 3 below.

Example 1: Archiving using Archive Push Technique:
The URL to issue to NG/AMS for a standard Archive Push Request could be something as:

http://hostx:7878/ARCHIVE?file_uri=/data/2002-02-11/File1.fits&wait=0[6]

In this case the client must also issue the data in the HTTP request. See also section 5.1. In this example the NG/AMS Server is instructed to send back an immediate reply, i.e., to reply before the request has been handled. This has the disadvantage that the client cannot know for sure if the request was successfully handled or not.

Example 2: Archiving using Archive Pull Technique:
The URL for the NG/AMS Server could be something like this:

http://hostx:7878/ARCHIVE?file_uri=ftp://hosty/data/2002-02-11/Fits1.fits

In this case the NG/AMS Server will pick up the file from the location given. I.e., the client need not to issue the data in the HTTP request. In the example shown, the NG/AMS Server will generate a reply after having handled the Archive Request.

[6] The values for the HTTP parameters must be encoded according to the specification of the HTTP protocol. Here they are shown not encoded, to make it more legible.

Example 3: Archiving from one NGAS System to NGAS System:
As a small 'curiosity', this example shows an Archive Pull Request, whereby the File URI specified is referring to a file located in an NGAS system:

http://hostx:7878/ARCHIVE?mime_type=application/fits&file_uri=http://hosty:7878/RETRIEVE?file_id=XYZ-2002-02-01T02:23:41.342

In this example, the NG/AMS Server handling the Archive Request will pick up the file from the remote NGAS host via a Retrieve Request, and will archive it.

19.2 RETRIEVE Command - Retrieving & Processing Files

The RETRIEVE command is used to retrieve archived data files from an NGAS system. The RETRIEVE command accepts the following parameters:

Parameter	*Mandatory*	*Description*
-process <DPPI>	No	With this parameter it is possible to specify a DPPI, which is invoked to process the data before sending it back. NG/AMS will send back the result of the processing, and not the original file. The parameter can be repeated to specify a list of DPPIs that which will be executed in the order specified.

It is possible to receive an HTTP redirection response as response to the Retrieve Request. In this case the client must re-send the Retrieve Request to the alternative URL given in the redirection response. See also section 5.3.

19.3 STATUS Command - Querying System Status & other Information

The STATUS command is used to query various status information from the NG/AMS Server. The STATUS command accepts the following parameters:

Parameter	*Mandatory*	*Description*
<no parameters>	No	In this case a reply is returned which contains an NG/AMS Status document. An example of such a status document can be found in section 2.16.
-disk_id <disk ID>	No	Query information about a disk referred to by its Disk ID. The reply is an NG/AMS Disk Status XML document. An example of this can be found in section 16.2.
-file_id <file ID>	No	Query information about a file with a given File ID. The reply is an NG/AMS XML Status document as shown in section 16.3.
-configuration_file	No	Query the configuration used by an NG/AMS server. The result is a complete NG/AMS Configuration XML document as shown in section 4.3.
-file_access <file ID>	No	This parameter is used to make an NG/AMS Server probe if it can physically access a file. The body of this request must contain an NG/AMS File Status XML document with the detailed information about the file. It is therefore necessary to issue also the "content-length" and "content-type" HTTP headers followed by the file status in the request. This command is thus similar to an Archive Push request, refer to section 5.1 for futher information about this issue.
-flush_log	No	Used to make the NG/AMS Server flush the logs it may have cached internally, into the Local Log File if such is specified.

It is only possible to specify one of the parameters at a time.

19.4 EXIT Command - Terminating Server

The EXIT command is used to make the NG/AMS Server exit. The EXIT command does not accept any parameters.

19.5 INIT Command - Re-Initializing the System

The INIT command is used make the NG/AMS Server re-initialize. This means that it will first go Offline, load the configuration and subsequently go Online. The INIT command does not accept any parameters.

19.6 LABEL Command - Generating Labels

The LABEL command is used to print out labels to be put on the disk cases. The label is the Logical Name of a disk. The LABEL command accepts the following parameters:

Parameter	*Mandatory*	*Description*
-slot_id < slot ID>	Yes	The ID of the slot in which the disk is installed.
-host_id <host ID>	No	The host in which the disk is installed. If this is not specified, the local hsot is assumed.

19.7 OFFLINE Command - Putting System Offline

The OFFLINE command is used to make the NG/AMS Server go Offline. The OFFLINE command does not accept any parameters.

19.8 ONLINE Command - Putting System Online

The ONLINE command is used to make the NG/AMS Server go Online. The ONLINE command does not accept any parameters

20 Index

G

H

I

J

L

M

N

O

P

Q

R

S